



## System Architect Essentials 8

Student Guide

© 2018 Pegasystems Inc.



© 2019  
**Pegasystems Inc., Cambridge, MA**  
**All rights reserved.**

## **Trademarks**

For Pegasystems Inc. trademarks and registered trademarks, all rights reserved. All other trademarks or service marks are property of their respective holders.

For information about the third-party software that is delivered with the product, refer to the third-party license file on your installation media that is specific to your release.

## **Notices**

This publication describes and/or represents products and services of Pegasystems Inc. It may contain trade secrets and proprietary information that are protected by various federal, state, and international laws, and distributed under licenses restricting their use, copying, modification, distribution, or transmittal in any form without prior written authorization of Pegasystems Inc.

This publication is current as of the date of publication only. Changes to the publication may be made from time to time at the discretion of Pegasystems Inc. This publication remains the property of Pegasystems Inc. and must be returned to it upon request. This publication does not imply any commitment to offer or deliver the products or services described herein.

This publication may include references to Pegasystems Inc. product features that have not been licensed by you or your company. If you have questions about whether a particular capability is included in your installation, please consult your Pegasystems Inc. services consultant.

Although Pegasystems Inc. strives for accuracy in its publications, any publication may contain inaccuracies or typographical errors, as well as technical inaccuracies. Pegasystems Inc. shall not be liable for technical or editorial errors or omissions contained herein. Pegasystems Inc. may make improvements and/or changes to the publication at any time without notice.

Any references in this publication to non-Pegasystems websites are provided for convenience only and do not serve as an endorsement of these websites. The materials at these websites are not part of the material for Pegasystems products, and use of those websites is at your own risk.

Information concerning non-Pegasystems products was obtained from the suppliers of those products, their publications, or other publicly available sources. Address questions about non-Pegasystems products to the suppliers of those products.

This publication may contain examples used in daily business operations that include the names of people, companies, products, and other third-party publications. Such examples are fictitious and any similarity to the names or other data used by an actual business enterprise or individual is coincidental.

This document is the property of:

Pegasystems Inc.  
One Rogers Street  
Cambridge, MA 02142-1209  
USA  
Phone: 617-374-9600  
Fax: (617) 374-9620  
[www.pega.com](http://www.pega.com)

# Course Objectives

After completing this course, you should be able to:

- Explain the benefits of using the Pega model-driven application design and development approach
- Identify the high-level responsibilities associated with Pega Platform for both Pega business architects and system architects
- Describe Pega's Direct Capture of Objectives approach to increasing the speed and accuracy of application delivery
- Model the life cycle of a case that mirrors the way business people think about how work is completed

## Course Objectives (continued)

- Validate case data to ensure that user entries match required patterns
- Configure a Wait shape to enforce a case processing dependency
- Configure user views and data elements during case life cycle creation
- Use the Clipboard tool to review case data in memory
- Set property values automatically using data transforms and declare expressions
- Configure and populate a work party with case data
- Create data types to model data objects in a Pega application

## Course Objectives (continued)

- Automate decision-making to improve process efficiency
- Design responsive user forms for use on any platform or browser
- Design reports to deliver key insights to business users
- Incorporate and manage reference data to allow applications to adapt to changing business conditions
- Delegate rules to the business to adapt cases types to changing business policies
- Test your application design to analyze rule behavior and identify configuration errors

# Become a Certified System Architect



## Certification Benefits

### Industry Credibility

The *Certified System Architect (CSA)* provides students with the recognition and credibility that demonstrates they know how to use Pega Platform to configure basic applications.



# GROUP DISCUSSION

© 2018 Pegasystems Inc.

8



## MODULE Introduction to Pega

This module contains the following lessons:

- Introduction to Pega
- Situational Layer Cake
- Application development studios
- Roles on a Pega project

Application developers need to be able to articulate tangible business and career benefits to developing applications with the Pega Platform so that they can efficiently master the development tool.

**After this lesson, you should be able to:**

- Compare Pega's application development approach to traditional application development
- List some of the major capabilities of the Pega Platform
- Identify ways you can develop and deploy Pega applications

# Pega's Application Development Approach

- Low-code / no-code
- Model Driven
- Visual Tools
- Business users work with IT

# Model-Driven Application Development

- Case types
- Cases
- Service Levels



© 2018 Pegasystems Inc.

4

# Major Features of Pega Platform

- Case Management
- Data Management & Integration
- Decision Management
- User Interface
- Reporting
- System Administration
- DevOps
- Mobility
- Security
- Robotic Automation
- Workforce Intelligence
- System Administration
- Intelligent Virtual Assistant
- Java and Activities

© 2018 Pegasystems Inc.

5

Community page: <https://community.pega.com/knowledgebase/capabilities>

# Pega Platforms Capabilities

- Direct Capture of Objectives (DCO)
- Guardrails
- Best practices
- Extensible applications
- Deploy on-premises or in the cloud



# GROUP DISCUSSION

© 2018 Pegasystems Inc.

7

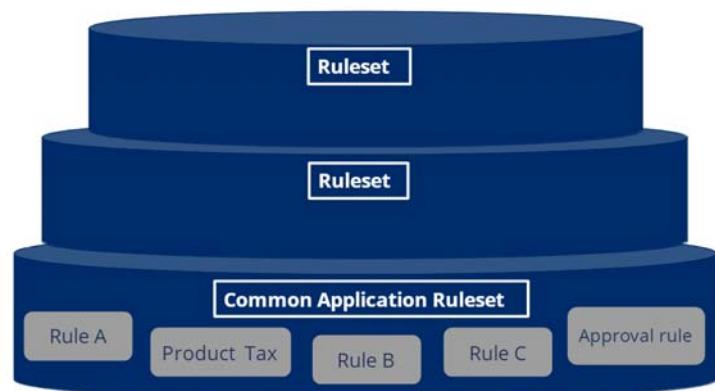
As an application developer, you can explain the Situational Layer Cake™ with an actual example that demonstrates that you know the business benefit of this feature.

**After this lesson, you should be able to:**

- Place high level assets in the appropriate layers of the Pega application structure

# Reusable Rules

- You create small reusable business functionality contained in **rules**.
- You create **rulesets** to group individual rules into containers for the purpose of deployment, versioning, and security.
- Pega calls this approach the **Situational Layer Cake**





## GROUP DISCUSSION

© 2018 Pegasystems Inc.

10

You can use a product tax example to discuss the layer cake.

Pega lets you create small reusable business functionality contained in rules. For example, consider a product tax calculation : Product Tax = Product Cost \* 10% Federal Tax. You configure this calculation in a rule called **Product Tax**.

You place this rule among many other rules that make up your application into rulesets which are containers that group individual rules for the purpose of deployment, versioning, and security.

You place this rule at a common (lower) layer of your situational layer cake which means it becomes the default Product Tax calculation.

Your roll out your first application in a specific region and your rule successfully executes appropriately calculating the product tax

You then plan to roll out your application to three additional regions two of which have various state or regional taxes.

Traditionally, a developer might add some if-then logic to calculate the tax for the different regions. As you roll out to more and more regions, this can quickly grow very complicated and add risk whenever you need to add or make changes to your tax calculations.

With Pega, you would simply create a new Product Tax calculation for each region that requires a unique regional calculation and need to do nothing to regions that have no special requirements.

You then place the unique regional calculations into new Regional rulesets where you group other application changes that make the specific regional application unique such as requesting additional data or approval processes for example.

You then roll out your application to the new regions and include the base layer as well as the regional specific layer and Pega will pick the most specific rule needed for your application. (Show for region A it picks the special Tax but for region B there is no special regional tax so it uses the default base layer rule. Show this in a cake model.)

Finally, show that all the different regions do run some common rules applicable to all such as loading product information. Should a change be needed here, it is done in one place at the common layer impacting all regional applications.

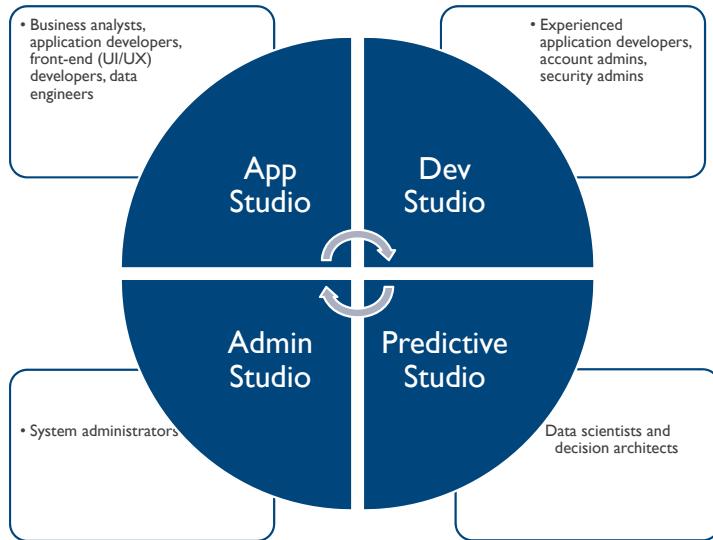
There is no limit to how many layers you can create making your situational layer cake as intricate as you need.

Application developer use four Pega Portals to perform development and system configuration tasks.

**After this lesson, you should be able to:**

- Identify the studios in Pega Platform
- Navigate between portals to perform system configuration tasks
- Identify representative tasks performed in each portal

# Pega Studios



© 2018 Pegasystems Inc.

12

# Studio Overview

- Studio highlights
- UI
- Access Groups
- Other Features

# App Studio

- Focuses on application development
  - Case Design
  - Data and integrations
  - Channels and interfaces (mobile, email, chatbots, etc.)
  - UI authoring
- Supports real-time UI design as you process cases (helpful for reviews with stakeholders)

# Dev Studio

- Focuses on advanced functionality
  - System settings
  - Complex rules - rule form access
  - Security
  - Component reuse (across studios)
  - Collaborative, branched development
  - Versioning and source control

# Admin Studio

- Focuses on system operations
  - Requestors
  - Jobs (agents)
  - Queue processors
  - Agents and queues
  - Listeners
  - Logs
  - APIs
- Supports both cloud and on-premise deployments

# Predictive Studio

- Focuses on defining AI models and assets
  - Predictive analytics
  - Adaptive analytics
  - Test analytics
  - Transparency policies
  - Data Exploration
- Areas – Predictions, Data, and Settings

# Access Groups and Default Studio

## Administrator

- Admin Studio

## Author

- App Studio

## Manager

- Case Manager Portal

## User

- Case Worker Portal



## GROUP EXERCISE

© 2018 Pegasystems Inc.

19



DEMO

## How to Navigate the Pega Platform

© 2018 Pegasystems Inc.

20

Pega projects involve different actors to achieve successful outcomes. In this lesson, you learn about the participants in a Pega project and how these participants work together to solve business problems.

**After this lesson, you should be able to:**

- List common roles on a project
- Match high-level tasks and responsibilities to the appropriate role
- Identify the business purpose of assembling a project team

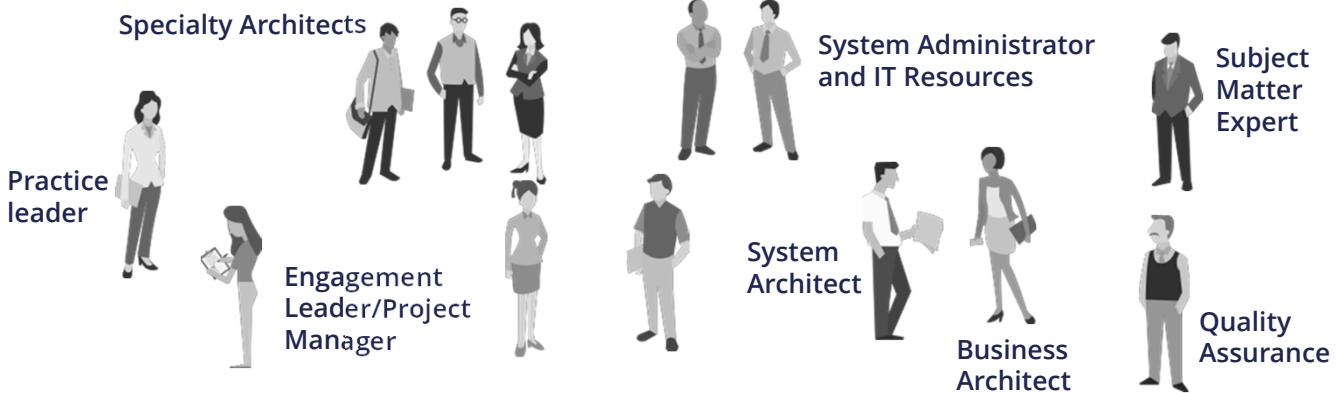
Pega projects involve different actors to achieve successful outcomes. In this lesson, you learn about the participants in a Pega project and how these participants work together to solve business problems.

**After this lesson, you should be able to:**

- List common roles on a project
- Match high-level tasks and responsibilities to the appropriate role
- Identify the business purpose of assembling a project team

# Members of a Project Team

- Successful projects require a good team with complementary skills
- The mix of team members and roles differs depending on the project goal



© 2018 Pegasystems Inc.

23

- **Business Architect (BA):** Work with subject matter experts and stakeholders to understand business needs. In a Pega application, they define business rules, service level agreements, and processes.
- **System Architects:** Application developers who design and configure the application. A lead or principal architect (LSA) designs the overall architecture while senior (SSA) and system architects (SA) configure assets such as user interface forms and automated decisions. They contribute object-oriented design and technical implementation skills.
- **Engagement Leader / Project Manager:** Provides overall project plan and delivery guidance.
- **Practice Leader:** Provides thought leadership, oversees project delivery and direction, establishes best practices and governance.
- **Product Owner:** Owns the product backlog and prioritization of backlog items. Creates acceptance criteria.
- **Quality Assurance (QA):** Creates and executes functional and performance test scripts, participates as part of the Scrum team.
- **Scrum Master:** Promotes and supports Scrum and ensures that all team members understand Scrum theory and practices. Facilitates Scrum events as needed.
- **Specialty Architects:** Engaged depending on project needs. For example, a UX Architect may be needed to design complex user views and a Deployment Architect may be needed to handle application deployment.
- **Subject Matter Expert (SME):** The SME has deep understanding of a particular business topic or domain. The SME works with the project team to convey business needs and helps validate information accuracy.
- **System Administrator and IT resources:** Provide expertise as needed. For example, on infrastructure, security, or integration.

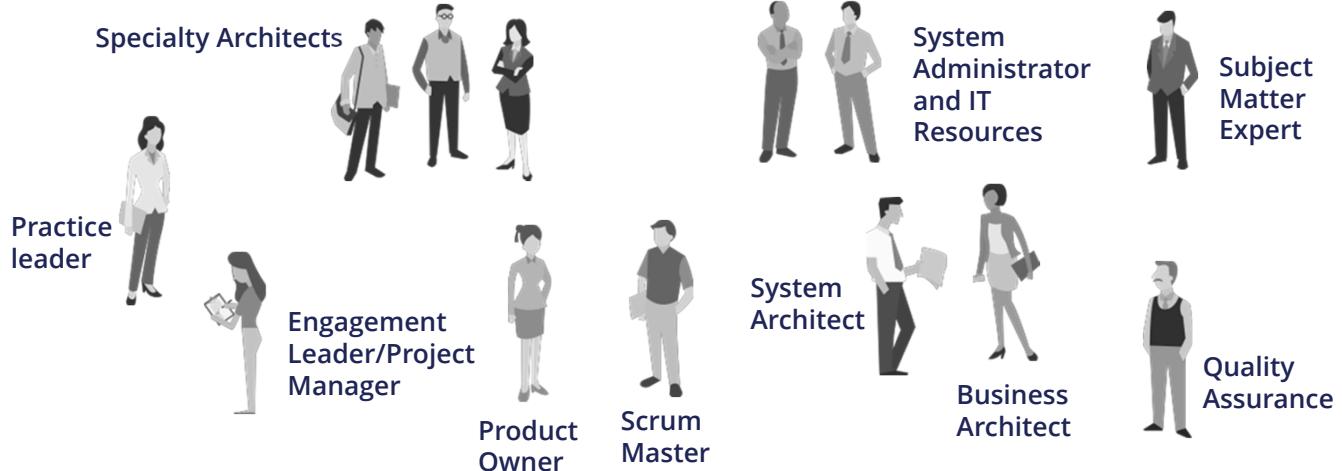
# Business Purpose of a Project Team

To deliver a successful Pega project, you need to assemble a project team that consists of members with the right mix of skills.



# Common Roles

The mix of team members and roles differs depending on the project goal. Some common roles found on a Pega project are described below.



© 2018 Pegasystems Inc.

25

- **Business Architect (BA)**: Work with subject matter experts and stakeholders to understand business needs. In a Pega application, they define business rules, service level agreements, and processes.
- **System Architects**: Application developers who design and configure the application. A lead or principal architect (LSA) designs the overall architecture while senior (SSA) and system architects (SA) configure assets such as user interface forms and automated decisions. They contribute object-oriented design and technical implementation skills.

## Other roles

- **Engagement Leader / Project Manager**: Provides overall project plan and delivery guidance.
- **Practice Leader**: Provides thought leadership, oversees project delivery and direction, establishes best practices and governance.
- **Product Owner**: Owns the product backlog and prioritization of backlog items. Creates acceptance criteria.
- **Quality Assurance (QA)**: Creates and executes functional and performance test scripts, participates as part of the Scrum team.
- **Scrum Master**: Promotes and supports Scrum and ensures that all team members understand Scrum theory and practices. Facilitates Scrum events as needed.
- **Specialty Architects**: Engaged depending on project needs. For example, a UX Architect may be needed to design complex user views and a Deployment Architect may be needed to handle application deployment.
- **Subject Matter Expert (SME)**: The SME has deep understanding of a particular business topic or domain. The SME works with the project team to convey business needs and helps validate information accuracy.
- **System Administrator and IT resources**: Provide expertise as needed. For example, on infrastructure, security, or integration.

# Citizen Developers

Citizen developers can also collaborate on Pega projects, bringing valuable knowledge about the business needs.



© 2018 Pegasystems Inc.

26

Citizen developers can also collaborate on Pega projects. **Citizen developers** are non-technical business users who participate in application development. These developers bring valuable knowledge about the business needs.

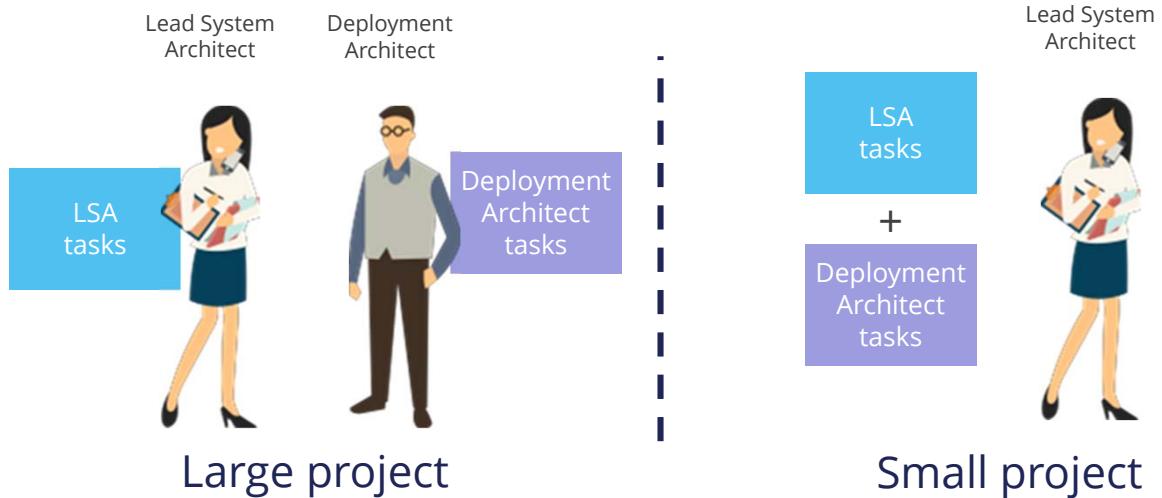
# Flexibility of Project Role Names

The role titles described by Pega may be referred to by different names depending on the organization implementing the project.



# Overlap of Roles Tasks

The size of a project affects how roles are distributed throughout the team.



© 2018 Pegasystems Inc.

28

The tasks and responsibilities for each role may overlap or be accomplished by multiple roles. For example, on a small project, a Lead System Architect may also perform the duties of the deployment architect.

# Co-Production

Pega recommends co-production to help customers gain proficiency, enabling them to support and expand their application moving forward.





## GROUP DISCUSSION

© 2018 Pegasystems Inc.

30

What roles are associated with the following responsibilities?

- Configure user interface forms (System Architect)
- Define business rules (Business Architect)
- Create acceptance criteria (Product Owner)
- Define service level agreements (Business Architect)
- Own prioritization of backlog items (Product Owner)



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

31

Summarize this module and respond to any questions.



This module contains the following lessons:

- Case life cycle
- User context

Business applications are the foundation of every organization. Business applications should function in the same way business users naturally think about and describe their work.

**After this lesson, you should be able to:**

- Identify the goal of a case life cycle design
- Define a case, a case type, a stage, a process, and a step
- Recognize the purpose of stage transition
- Design a case life cycle

# Case Life Cycle Design

Business applications help automate work that is required to achieve specific business outcomes.

Pega thinks applications should function the same way the user think about and describe their work.



# Case and Case Type

- A **case** is work that delivers a meaningful business outcome.
- A **case type** is an abstract model of a business transaction, where a case is a specific instance of the transaction.
- In a Pega application, you use case types to model repeatable business transactions.

# Stages

The **case life cycle design** model allows business users to begin by organizing work into stages.



**Stages** represent the transfer of a case from one authority to another or a significant change in the status of the case.

A **primary stage** is a high-level phase in the lifecycle of a case that leads to the desired outcome.

# Organizing and Naming Steps

- Use a **noun** or **noun phrase** to describe the context of the section you are in.
- As much as possible, try to use no more than two words.
- Use names that are more meaningful and relevant to the business users.
- Consider limiting the number of steps in each process to **five, plus or minus two**.
- When naming processes and steps, use a **verb + noun** naming convention.

# Process Steps

A process (for example, **Place Order**, **Receive Order** and **Ship Items**) contains a series of tasks, or steps, that a user completes as they work the case.

A **step** (for example, **Pack Items** or **Deliver to Customer**) is either a user action or automated action performed by the application.





# GROUP DISCUSSION

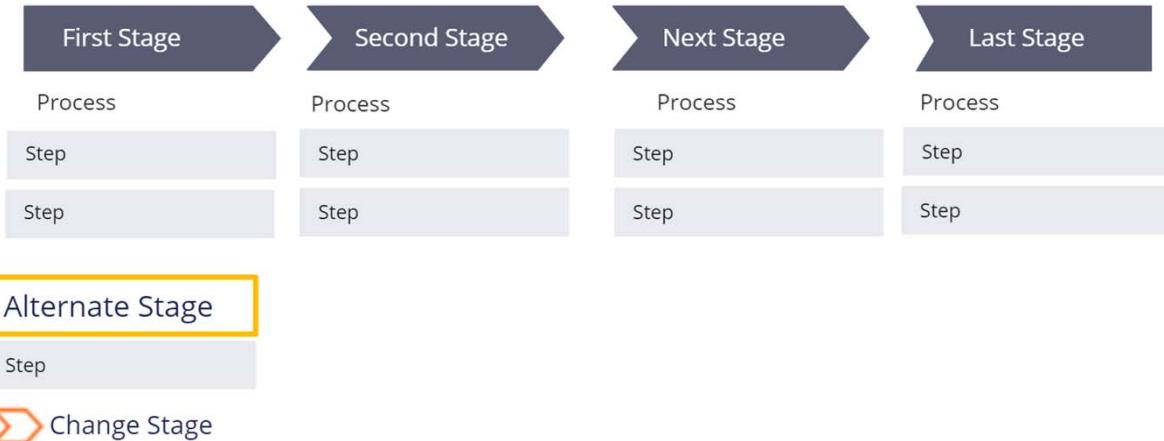
© 2018 Pegasystems Inc.

8

# Stage Transitions

- For **primary stages**, when all the steps in a stage are complete, the default option is an automatic stage transition to the next primary stage.
- When a case does not follow the primary path, use **alternate stages** to organize process steps.
- A **Change Stage** step automates non-sequential case flow, such as to and from alternate stages.

# Stage Transitions





DEMO

## How to Design a Case Life Cycle

© 2018 Pegasystems Inc.

11

The case IDs by themselves do not help to determine what work you need to perform on those cases. The case status and instructions help identify the state of the case and provide user context.

**After this lesson, you should be able to:**

- Identify the role of case status in case processing
- Identify when the status of a case updates
- Identify the role of instructions in assigning tasks to users

# Updating the Case Status

- The **case status** is the primary indicator of the progress of a case towards resolution.
- The case status is updated as the case moves through the case life cycle.
- You can set the case status on each step or stage in the case life cycle.
- Pega includes standard case status values such as **Open**, **Pending-Approval**, and **Resolved-Completed**.
- You can also add custom status values.



DEMO

## How to Update Case Status

© 2018 Pegasystems Inc.

14

# Adding Instructions

- An **instruction** for a step identifies to a user what should be accomplished in an assignment.
- Users see instructions in their worklist, when they open a case, or when they are prompted for input in a user view.



DEMO

## How to Add Instructions to Assignments



# MODULE Setting a Service Level

This module contains the following lessons:

- Service level agreements

Organizations often establish service level agreements to enforce on-time performance. These obligations range from informal promises of response times to negotiated contracts.

**After this lesson, you should be able to:**

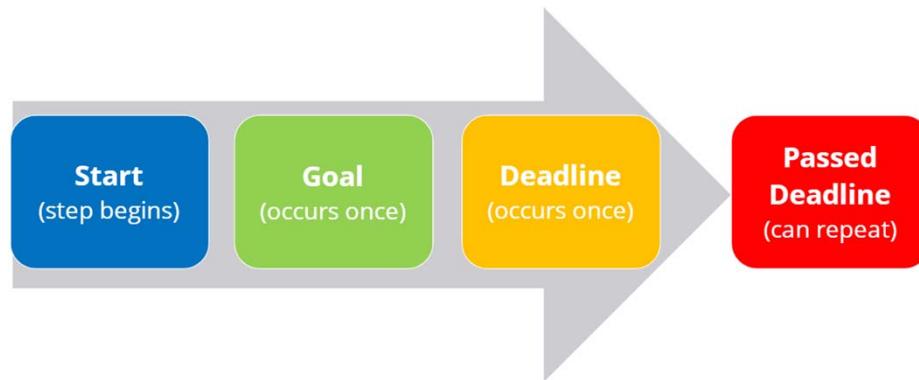
- Differentiate between the three service level intervals
- Identify the four different objects where the service levels can be applied
- Identify the business values of urgency and examples of how it is used in a Pega application
- Calculate urgency when one or more service level intervals occur

# Service Level Agreements

- A **service level agreement (SLA)** establishes a deadline for work completion.
- When you establish a goal and deadline, Pega creates service level agreement rule for you.
- Service levels use three milestones to indicate the expected turnaround times for the assignment or the overall case on which they are defined.

# Service Level Intervals

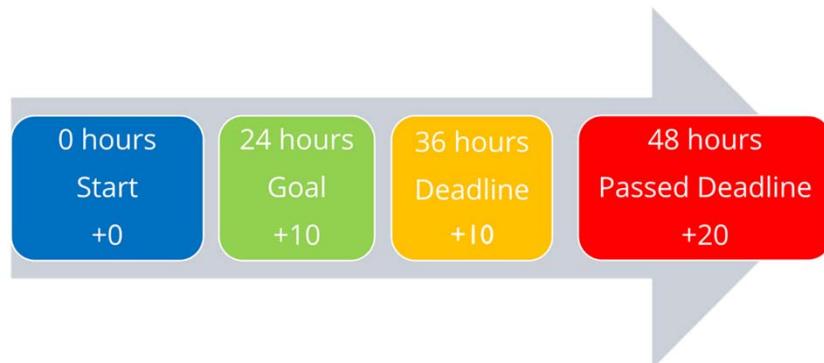
You can configure service levels for steps, processes, stages, and entire cases.



# Defining Urgency

You define an urgency between 10 and 100 for each milestone.

The higher the value, the higher the urgency. Typically, the urgency increases as an assignment advances to the next milestone.



# Assignment Priority

- In Pega applications, assignment urgency indicates assignment priority.
- The **Get Next Work** functionality of a Pega Platform assigns high urgency tasks before low urgency tasks to ensure the assignments are completed in a timely manner.
- Between two or more assignments that users can perform, Get Next Work favors the assignment with the highest(greatest) urgency.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

7

How does the passed deadline interval differ from goal and deadline intervals?



DEMO

## How to Add a Service Level to an Assignment



DEMO

## How to Add a Service Level to a Case



DEMO

## How to Apply Service Levels Throughout the Case Life Cycle

© 2018 Pegasystems Inc.

10



This module contains the following lessons:

- Optional actions
- Skipping processes and stages
- Parallel processing
- Decision points

Optional actions enable users to leave the primary path of a case to complete another task or process. The actions supplement the tasks users can do as they work on a case.

**After this lesson, you should be able to:**

- Explain the role of optional actions in a case
- Differentiate between optional user actions and optional processes

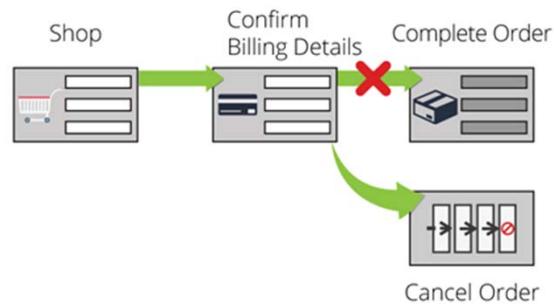
# Optional Actions to a Case Type

- You define optional actions that users can perform while a case is in any stage to step of the life cycle.
- By allowing users to choose when additional processing is needed, you can support out-of-sequence events in a case.
- You can configure two types of optional actions:
  - As a **process**
  - As a **user action**

# Optional Process

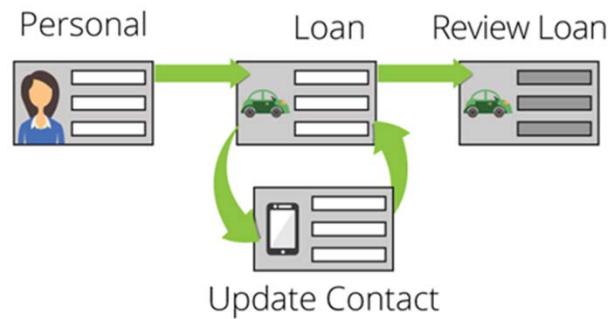
Users can launch an **optional process** to update information in multiple steps.

Users may or may not return to the primary path of the case.



# Optional User Actions

**Optional user actions** update information in a single user screen.



# Adding Optional Actions to a Case Type

- You can make an optional action available anywhere in a case life cycle or in a specific stage.
  - Case wide actions allow actions throughout the case.
  - Stage-only actions allow actions during that stage.



DEMO

## How to Add Optional Actions to a Case



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

8

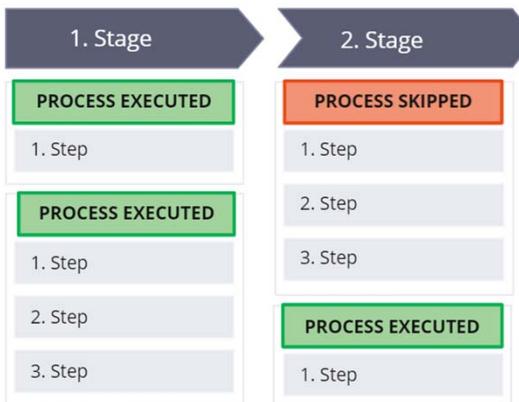
Addressing when to use an optional user cation and when to use an optional process.

An automobile accident case contains a process to open a medical claim if a party was injured in the accident. However, if the car was hit while parked and unattended, the medical claim process can be skipped.

**After this lesson, you should be able to:**

- Configure a process or stage to run when certain conditions are true

# Skip a Process or Stage



- You can define conditions that control whether a process or stage runs in a case by using comparators.
- If any of the conditions match the comparators, the process or stage is skipped.

## Conditions

When	Field	Comparator
	Remote employee? ▾	is false ▾

© 2018 Pegasystems Inc.

10

# Skipping Process or Stage

- By default, a **process** starts and skips only when a condition is present.
- By default, a **stage** never skips unless a condition is present.

**Note:** Skipping either a stage or a process can be configured in App Studio.

# Number of Conditions

You can add more than one condition to a stage or process.

You can click the add icon to logically join two or more conditions using the **AND** or **OR** operators.

The screenshot shows a user interface for defining conditions. At the top, there's a header labeled "Conditions". Below it, there are three main fields: "Field" (set to "Personal injury?"), "Comparator" (set to "is true"), and "Value" (set to "50"). To the right of these fields is a yellow-bordered "+" button. Below these fields, there's a dropdown menu with three options: "AND", "AND", and "OR". The "AND" option is currently selected and highlighted in blue. To the right of this dropdown are two more buttons: a blue-bordered "+" button and a trash bin icon. The entire interface is set against a white background with a thin gray border.



DEMO

## How to Skip a process or Stage in a Workflow



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

14

How would a case workflow look when we have to skip a process or stage?

A new hire onboarding case contains an IT Setup process to provide a configured laptop to the new employee and a Facilities Setup process to assign an office to the new employee. These processes can run in parallel.

**After this lesson, you should be able to:**

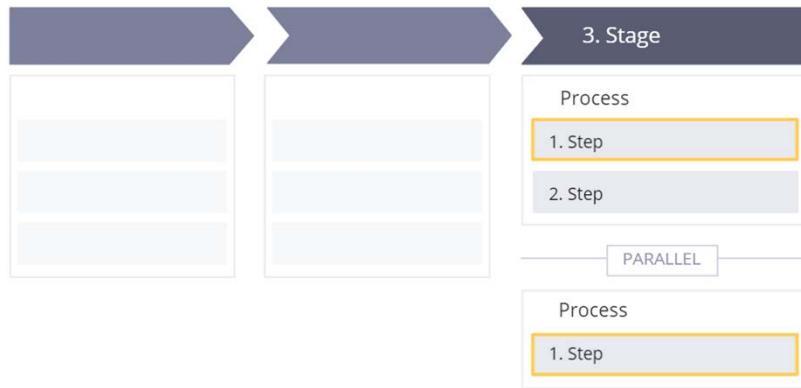
- Identify a use case for parallel processing
- List the business benefits of parallel processing

# Parallel Processes

- If processes can be performed in any order, they can be configured as **parallel processes**.
- Parallel processes allow a case to advance through multiple paths at the same time within a stage.
- Parallel processes are created in Dev Studio.
- You can configure two or more processes to run in parallel.

# Efficient Case Processing

When the case is processed, the active assignment in either process can be performed. This increases the efficiency of case processing.





DEMO

## How to Perform Processes in Parallel



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

19

When would you go for parallel processing while design your case workflow?

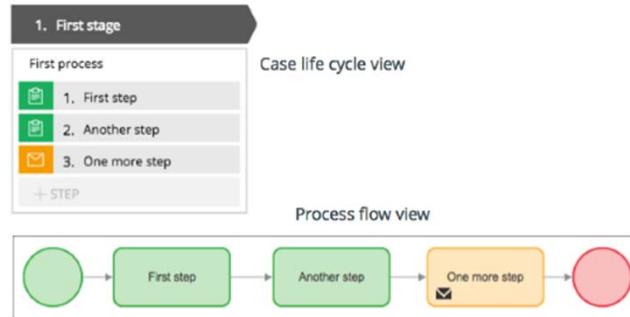
Use decision shapes to model automated decisions. Define automated decisions by a set of one or more conditions or business logic to evaluate.

**After this lesson, you should be able to:**

- Describe how decision points affect case processing
- Describe use cases for automating decisions using decision shapes

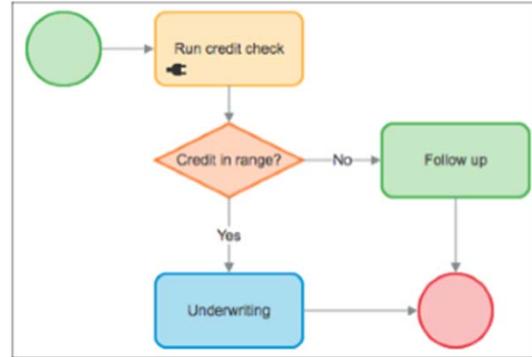
# Case Life Cycle Workflow

The case life cycle workflow defines the order in which tasks must be executed.



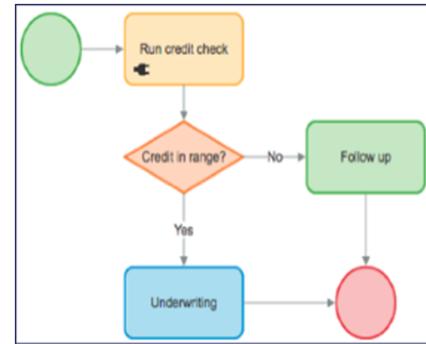
# Decisions Shapes

- Decision shapes are based on a when condition and are configured to advance a workflow automatically.
- For example, when a home is above a certain age, it must be tested for lead paint.
- When a home is below a certain age, it does not need to be tested for lead paint.



# Modeling Automated Decisions

- For example, the decision shape *Credit in range?* models an automated decision. This will determine if an insurance agency wants to sign a new policy holder. If the result is *Yes*, then the customer is notified and the work is routed to the underwriter. If the result is *No*, then the case proceeds to follow up with the customer.





## GROUP DISCUSSION

© 2018 Pegasystems Inc.

24

A prospective employee must pass a criminal background check before receiving an offer. When would you add a decision point to the workflow process?



## MODULE Routing Work to Users

This module contains the following lessons:

- Routing work
- Case approval configuration

You can assign a task to a user's worklist or to a work queue when you add an assignment step and an Approve/Reject step. You can also leverage automated routing capabilities that enable you to route tasks based on criteria that you specify.

**After this lesson, you should be able to:**

- Articulate the business value of routing
- Provide a business use case for routing
- Distinguish a worklist from a work queue
- List some of the available out-of-the-box routing options

# Define Who Does the Work

While modeling a process, you define who should do the work on each assignment.

It is common for more than one person (**operator**) to complete work on a case.

Careful and appropriate assignment routing design increases business efficiency because assignments go to the individual or group of individuals most capable of completing a specific assignment.



# Routing Types – Work Queue

- You use **assignment routing** to assign work to the most appropriate user. There are two types of routing — work queue and worklist.
- A **work queue** is a list of all open assignments, in order of importance, for a group of users.
- Assignments stay in the work queue until a user associated with the work queue selects an assignment, or a manager sends an assignment in the work queue to a specific user.

# Routing Types – Worklist

- A **worklist** is a list of all open assignments, in order of importance, for a specific user.
- For example, an assignment that requires a human resources manager to approve employee time off requests routes to the worklist of the human resources manager.

# Routing Options

- You route an assignment to the **current user** if the current user should perform the task. For example, the employee creating the request enters the expense details.
- You route an assignment to the work list of a **specific user** if a specific user will complete the assignment. For example, the manager who approves expense reports.
- You route to a **work queue** for a specific group when anyone in the group can complete the assignment. For example, the payroll department work group that creates and sends the payment to the employee.

# Operator Record

In Pega Platform and applications, users are associated with operator records. The operator record contains information useful for routing.

Record type
Organizational structure
Work group
Work queue
Skill
Calendar
Scheduled absence

# Operator Record Example

Example operator record configuration for an employee in the payroll department:

Record type	Payroll employee
Organizational structure	<i>Payroll department in the European Retail division</i>
Work group	<i>EurRetailPayroll</i>
Work queue	<i>EurRetailPayroll/Expenses</i>
Skill	<i>ExpenseReimbursement</i>
Calendar	<i>European calendar</i>
Scheduled absence	<i>December 26 – January 4</i>

# Business Logic Routing

You use business logic routing based on a **when condition** when you want to route work based on certain conditions.

The screenshot shows a configuration interface for business logic routing. At the top, a header reads "Business logic" and "Route work based on these conditions". Below this, there are two main sections: "1" and "otherwise".

- 1:** Contains an "Action" dropdown set to "Route to operator" and a "Value" input field.
- When:** Contains a "Field" dropdown set to "Please select a field.", a "Comparator" dropdown, and a "Value" input field.
- + Add condition:** A link to add another condition.
- otherwise:** Contains an "Action" dropdown set to "Route to operator" and a "Value" input field.

# Out-Of-The-Box (OOTB) Routing Options

- Pega Platform provides an extensive list of routing options. The options covered so far include the following:
  - Work queue of a work group
  - Worklist of a specific user
  - Current user
  - Business logic based on a when condition
- You can also configure routing using specialized routers, such as to a skilled group, a workgroup manager, and a party to a correspondence.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

11

How does assignment routing improve efficiency?



DEMO

## How to Route Work

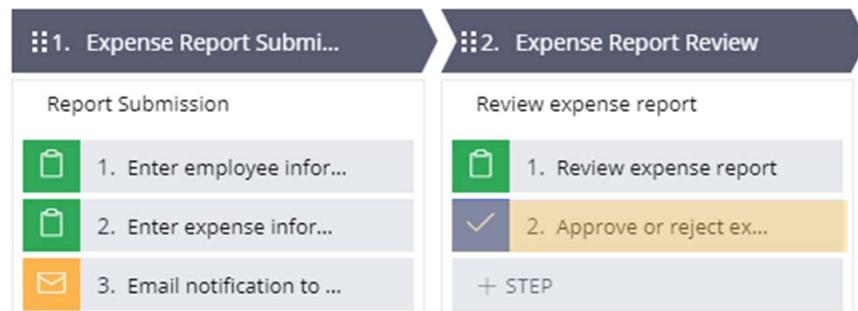
Approvals vary depending on the case type. An application for auto insurance may need one approval from the company underwriter. Some case types, such as a purchase request or an expense report, may need a series of approvals.

**After this lesson, you should be able to:**

- Articulate the business purpose of the Approve/Reject shape
- Distinguish between the types of approvals supported by the Approve/Reject shape
- Identify an appropriate approval configuration to satisfy business requirements

# Case Approvals

Case approvals are decision points at which one or more users decide whether to approve or reject a case.



# Approve/Reject Considerations

When configuring an approval, consider the following:

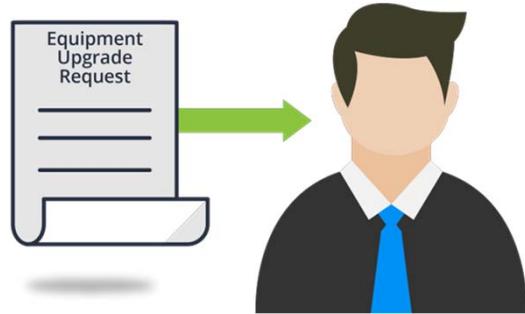
- Who needs to perform the approval: a single user or multiple users?
- For a cascading approval, how many approvals are required — is approval by the entire reporting structure required, or a subset of the hierarchy? For example, director approval is required only for expenses reports that exceed a specified total.
- How are approvals and rejections handled? Should the case change to a different stage, or can processing continue? Is a status change needed?

# Approval Types

- There are three types of approval types supported by the **Approve/Reject** shape:
  - **Single approver**
  - **Cascading approval – reporting structure**
  - **Cascading approval — authority matrix**

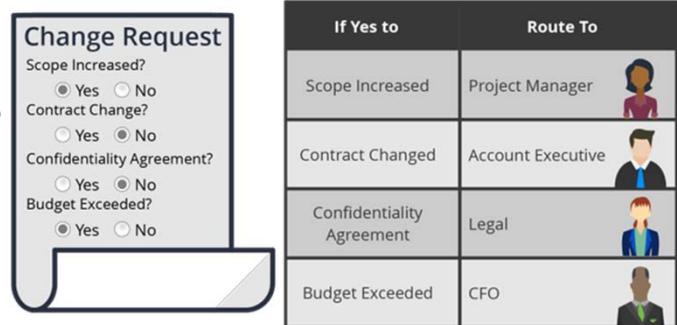
# Use Case – Single Approval Use Case

- You can assign single approvals to the worklist of a specific user or a work queue.
- For example, if an account manager can approve a request for an equipment upgrade, the approval goes to the manager's worklist. If any member of a group of managers can approve the request, the approval goes to the group's work queue.



# Use Case – Cascading Approval, Authority Matrix

- A cascading approval based on an authority matrix is more flexible than reporting structure. It supports routing to other entities outside of the reporting structure.
- For example, a change request for a consulting project needs various approvals based on the type of change.



## Use Case – Cascading Approval, Authority Matrix

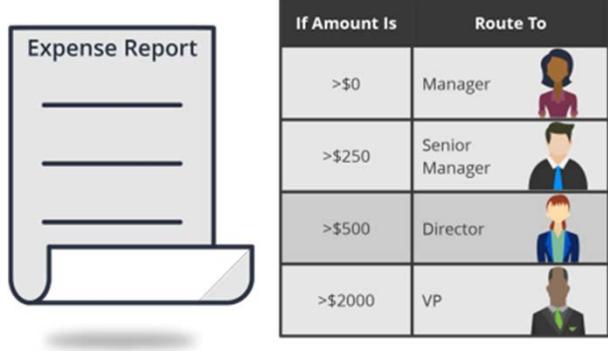
- Authority matrixes need additional configuration, including a data structure identifying the approvers and a property to identify the current approver.

## Use Case – Cascading Approval, Reporting Structure

- Cascading approvals based on reporting structure require approval of an employee's direct manager and higher. You can also configure business logic to set thresholds to determine the number of required approvals.
- For example, an expense report requires manager approval before accounting can process the payment. Depending on the total of the expenses submitted, the expense request may also need approval by a senior manager, a director, and a vice president.

# Use Case – Cascading Approval, Reporting Structure

- The reporting structure approval generates a set of approvers using information on the user's operator ID record: either the reporting manager or the manager of the workgroup to which the user is assigned.
- At runtime, Pega Platform looks up the appropriate approver based on the current user.





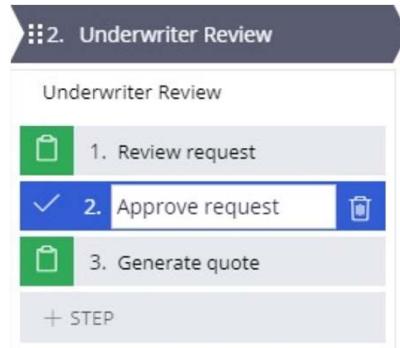
## GROUP DISCUSSION

© 2018 Pegasystems Inc.

22

# Approve/Reject Shape

- You configure approvals in Pega Platform case type work flows using the **Approve/Reject** shape.
- The Approve/Reject shape calls the process that you configure to manage the case approvals.
- Approvals generally continue the case. You can configure rejections to automatically resolve a case.



# Approve/Reject Shape Process

- You configure the consequences of approval and rejection in the contextual properties panel.
- You can also change the case status when the assignment is approved.
- The process you configure applies to all approvers of the specific assignment.

# Cascading Approval – Reporting Structure

You configure cascading approvals in Dev Studio.

- Reporting structure cascading approval options include:
  - The operator that completes the approval (Reporting manager or Workgroup manager)  
**One** – The reporting manager or the workgroup manager  
**All** – The entire manager hierarchy  
**Custom** – Number of levels as needed to evaluate when rules



DEMO

## How to Configure a Single Level Approval



DEMO

## How to Configure Cascading Approvals



## MODULE Configuring a Case Hierarchy

This module contains the following lessons:

- Rules and rules types
- Rules and rulesets
- Classes and class hierarchy
- How to update a rule
- How to reuse rules through inheritance
- Case hierarchy
- How to enforce a dependency between case types

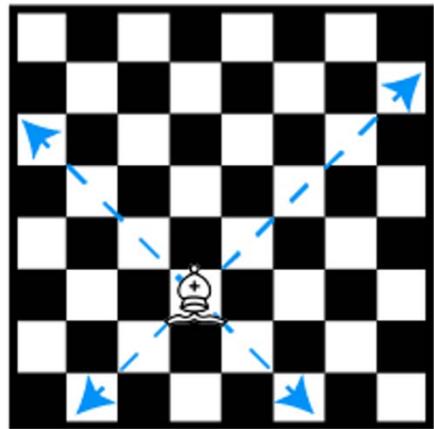
When you play a game of chess, you and your opponent agree to follow a specific set of instructions. These basic instructions are the rules of chess.

**After this lesson, you should be able to:**

- Describe the relationship between an application and rules
- Differentiate between a rule and a rule type

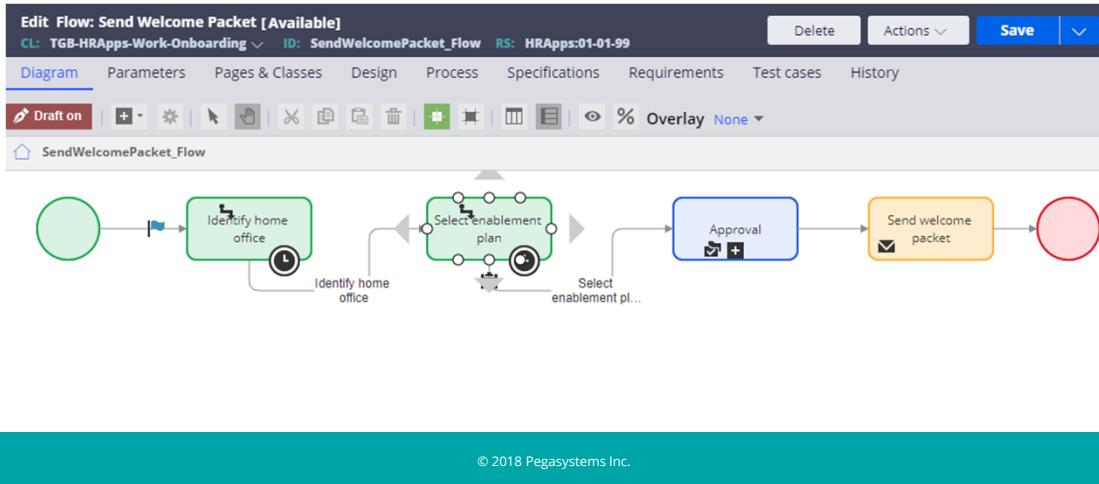
# Rules

- Like chess, Pega cases are governed by a set of rules.
- **Rules** describe the behavior of individual cases.
- A **rule type** is an abstract model of a specific case behavior. Rules are instances of rule types which generate application code.
- Pega creates many rules through wizards.  
The process, data elements, UI forms created by App Studio and Dev Studio are rules.



# Rule Types

Each rule is an instance of a rule type. You can think of a rule type as a template for creating a rule.



© 2018 Pegasystems Inc.

4



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

5

What is the purpose of a rule in a Pega application?

If a rule is similar to a song, a ruleset is similar to an entire album. Just as you can copy the album to share with a friend and allow your friend to listen to your favorite song, you can share a ruleset between applications to allow several applications to use the same rules.

**After this lesson, you should be able to:**

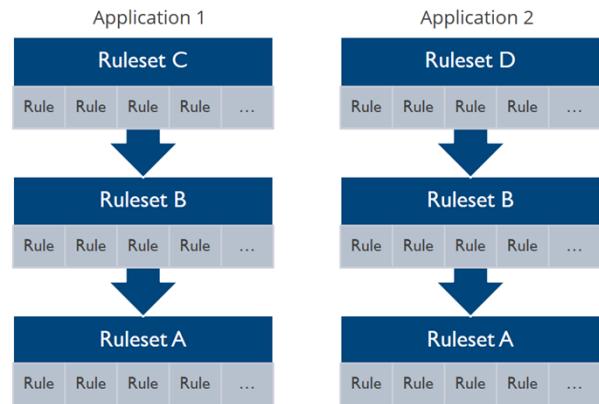
- Rules and rulesets
- Ruleset Versioning
- Ruleset Stack

# Rules: Modular Instructions

Individual rules make application behavior modular, providing versioning, delegation, and reuse capabilities.

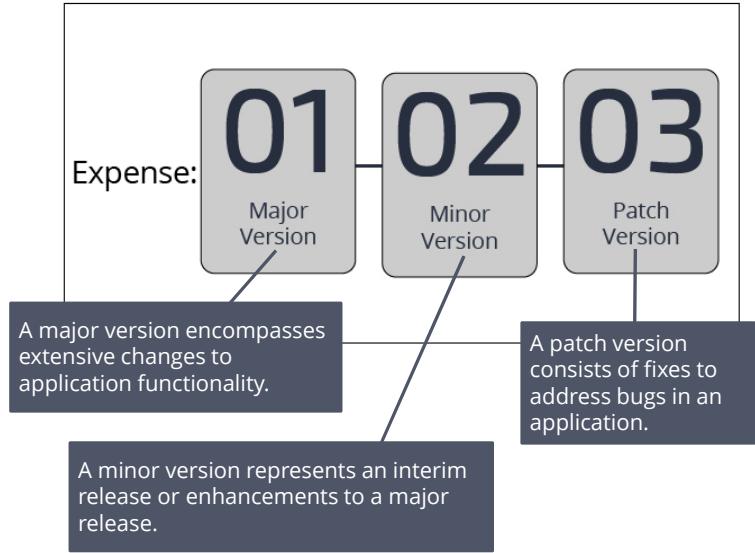
# Rules and Rulesets

- Each rule describes a specific facet of case behavior.
- **Rulesets** package rules for distribution as part of an application.
- Rulesets can be shared between applications.



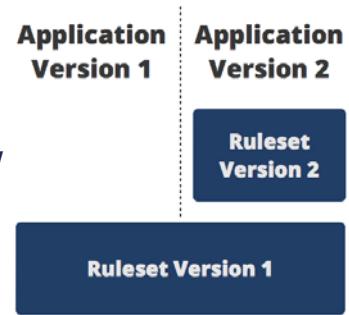
# Ruleset Versioning

- System architects collect individual rules into a subset of a ruleset, called a **ruleset version**.
- To update the contents of the ruleset, create a new ruleset version.
- Ruleset versioning enables easy application updates.



# Ruleset Stack

- Each application consists of a sequence of rulesets, called a **ruleset stack**.
- The ruleset stack determines rule priority.
- This allows updates to applications by referencing new ruleset versions.
- Example: Expense report application
  - Bob works on the first version and creates Expense:01-01-01.
  - Tanya receives an enhancement request and creates Expense:01-02-01.





## GROUP DISCUSSION

© 2018 Pegasystems Inc.

11

A ruleset version is identified with a string of three numbers. What do these three numbers indicate?

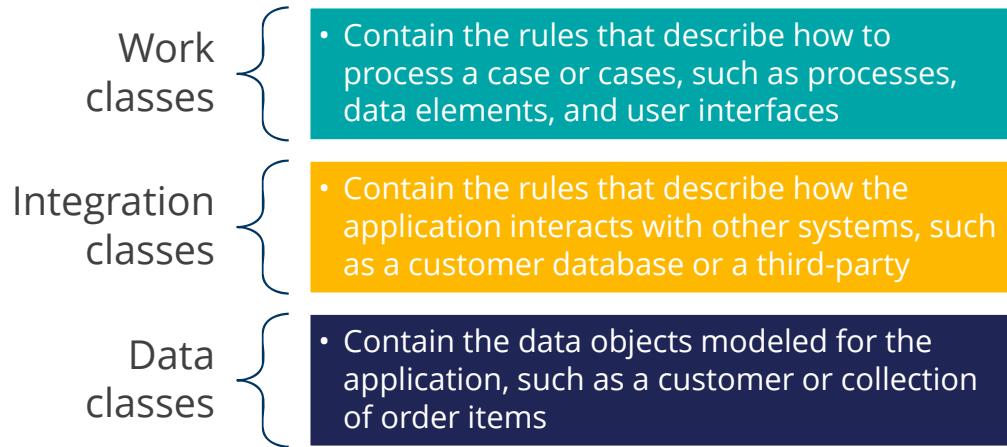
One strength of the Pega Platform is the reuse of rules between case types and applications. System architects often reuse rules—from single data elements to entire processes—in applications.

**After this lesson, you should be able to:**

- Types of classes
- Parent/child classes
- Class hierarchy

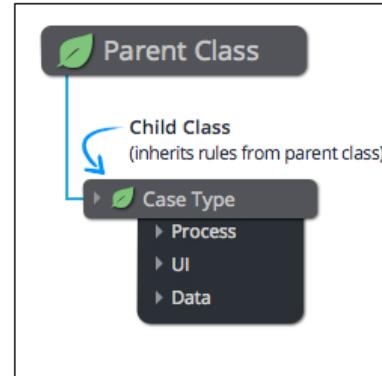
# Types of Classes

Pega platform supports reuse of rules between case types and applications. Each application consists of three types of classes.



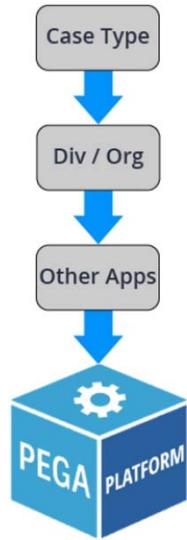
# Parent/Child Classes

- A class can contain other classes.
  - A **parent** class contains another class. A **child** class is contained by another class.
  - A child class can reuse, or inherit, any of the rules defined for its parent class.



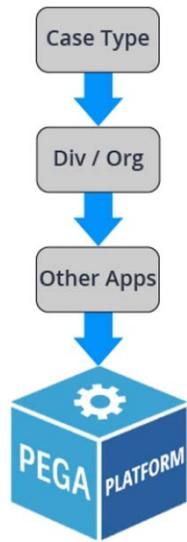
# Class Hierarchy

- Classes are organized into a class hierarchy.
  - The class hierarchy dictates rule reuse within the application.
- The class hierarchy consists of several groups of classes:
  - Classes that describe a specific case type, such as insurance claims
  - Classes that collect common rules and data elements
  - Classes from other applications, such as industry-specific Pega apps
  - Base classes provided by the Pega platform



## Class Hierarchy (continued)

- Any rule available to an application through the class hierarchy is considered “in scope”, otherwise they are considered “out of scope”.
- The class name indicates the position of the class within the hierarchy.
  - Each level of the class hierarchy is separated by a hyphen (-).
  - Example: The class *TGB-HR-Work* has the following parent classes:
    - *TGB-HR*
    - *TGB*





## GROUP DISCUSSION

© 2018 Pegasystems Inc.

17

What is the purpose of a class in a Pega application?



DEMO

## How to Create a New Rule

System architects often secure rulesets to prevent unauthorized or unintended changes to rules. When you edit the rules in a secured ruleset, you either check out the rule or perform a private edit.

**After this lesson, you should be able to:**

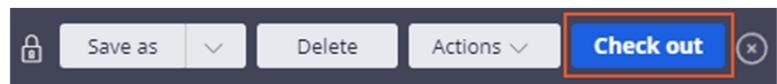
- Update a Secured Rule
- Check-out from a rule
- Perform private edit on a rule

# How to Update a Secured Rule

- System architects often secure rulesets to prevent unauthorized or unintended changes to rules.
- When you edit the rules in a secured ruleset, you either **check out** the rule or perform a **private edit**.

# Rule Check-Out

- Rule check-out creates a copy of a rule in a ruleset that is only visible to you, called a **personal ruleset**.
  - After you update the rule, you **check in** the rule, which updates your application.
- The personal ruleset occupies the top spot in the ruleset stack.
  - The rules in your personal ruleset override rules in the rest of the application.
  - This allows you to test your changes to the rule without affecting other system architects.



# Private Edit

- A private edit provides a nonexclusive check out of a rule.
- This allows other system architects to edit a rule at the same time.
- Private edits are useful for quick debugging without interrupting development by other team members.
- Best practices:
  - Lock older versions of a ruleset to prevent changes; lock icon appears.
  - To update a rule in a locked ruleset version, save the rule to an unlocked ruleset version.



Inheritance allows your application to reuse rules that have already been created for other cases or applications.

**After this lesson, you should be able to:**

- How to reuse rules through inheritance
- Sample inheritance diagram

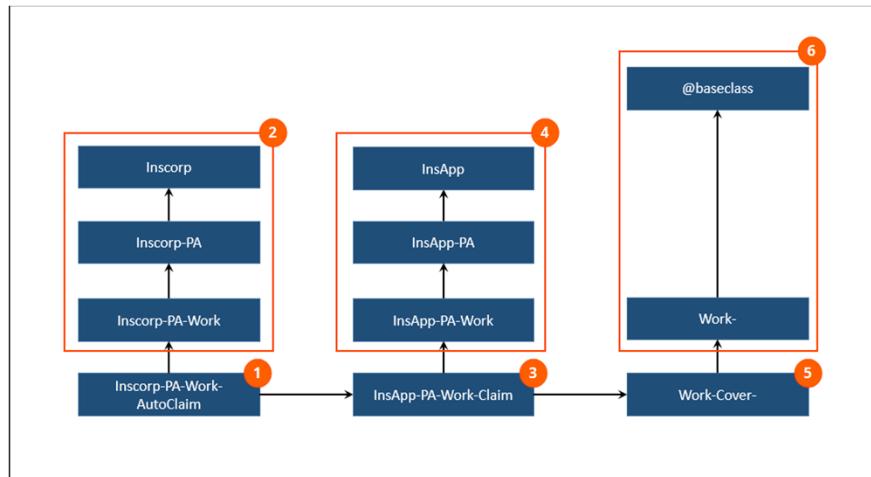
# How to Reuse Rules Through Inheritance

- Inheritance allows your application to reuse rules that have already been created for other cases or applications.
- Reduces development and testing time without sacrificing application quality.
- Pega provides two methods for inheriting rules:
  - **Pattern inheritance** follows the class hierarchy to support the reuse of rules within an application.
  - **Directed inheritance** allows your application to reuse rules from other applications with different classes.

Rule reuse through inheritance in Pega:

1. First searches pattern inheritance
2. If unsuccessful, uses directed inheritance
3. Repeats until @baseclass
4. Returns an error if no rule found

# Sample Inheritance Diagram



An auto claim inherits a data element for the case ID, found in @baseclass.

Consider the following example, in which an auto insurance claim case references the data element that stores the case ID. This data element belongs to the ultimate base class, `@baseclass`. The application containing the auto insurance claim is built on a generic policy administration application, which is built upon the Pega platform.

1. An auto claim case, described by the class `INSCORP-PA-WORK-AUTOCLAIM` references the case ID data element.
2. The data element is not found in the class `INSCORP-PA-WORK-AUTOCLAIM`, so Pega searches through the parent classes using pattern inheritance.
3. The data element is not found though pattern inheritance, so Pega searches the parent class specified by directed inheritance, `INSAPP-PA-WORK-CLAIM`. This class belongs to the generic policy administration application.
4. The data element is not found in the class `INSAPP-PA-WORK-CLAIM`, so Pega searches its parent classes using pattern inheritance.
5. The data element is not found though pattern inheritance, so Pega searches the parent class specified by directed inheritance, `WORK-COVER-`. This class belongs to the Pega platform.
6. The data element is not found in the class `WORK-COVER-`, so Pega searches its parent classes using pattern inheritance, finally locating the data element in `@BASECLASS`.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

26

What type of relationship is described by pattern inheritance?

How does directed inheritance differ from pattern inheritance?

From which class does @baseclass inherit rules?



DEMO

## How to Update a Rule

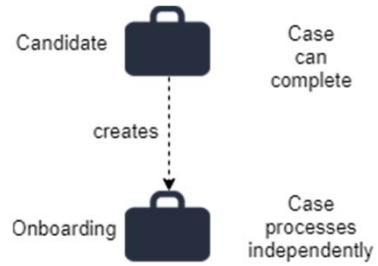
A business transaction can be complicated and involve multiple cases, where some maybe independent to each other and some dependent on each other.

**After this lesson, you should be able to:**

- Case hierarchy
- Differentiate between Top-level and Child case

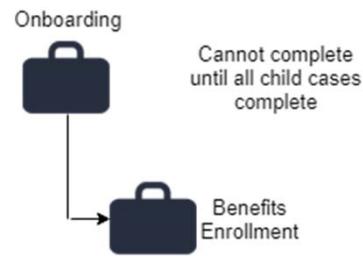
# Case Hierarchy

- Consider the new hire process. During the interview process, the human resources (HR) department opens a Candidate case for each job applicant. If the interview is successful, the applicant receives a job offer.
- When the candidate accepts the job offer, HR considers the candidate hired and is now an employee. The Candidate case is completed and creates an Onboarding case to prepare for the new employee's start date.
- In this example, the Onboarding case is independent of the Candidate case.



## Case Hierarchy (continued)

- Now before an Onboarding case can be approved the Benefits Plan case must be resolved. In this example, the Onboarding case and the Benefits Enrollment case have a case-subcase relationship.
- The system associates the Benefits Enrollment information with the Onboarding case. This allows you to join this associated information when reporting or auditing Onboarding cases.



# Differentiate between Top-level and Child Case

In a Pega application, you can model this case-subcase relationship with a case hierarchy that contains a top-level case and child case.

- Top-level – A case that does not have a parent case, but can cover, or become a parent of, other cases.
- Child – A case that is covered by a parent case. When you configure a case as a child case, Pega maintains a relationship between the parent and child cases. Child cases represent work that must be completed to resolve the parent.



You can enforce dependencies between parent and child cases with the Wait step.

**After this lesson, you should be able to:**

- Enforcing a dependency between case types

# Enforcing a Dependency between Case Types

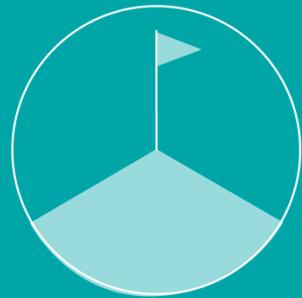
- You can enforce dependencies between parent and child cases with the Wait step. When a parent case reaches the Wait step, the case pauses until the dependency resolves. Once the dependency resolves, the case resumes processing.
- The Wait step enforces the following types of dependencies:
  - Pausing a case until another case (or all cases) reach a specified status
  - Pausing a case until a predetermined time expires



DEMO

## Adding a Child Case to a Case





## CAPSTONE EXERCISE

### Configuring a Case Type

© 2018 Pegasystems Inc.

## MODULE Adding Fields to a Case Type

This module contains the following lessons:

- Data elements in Pega applications
- Managing properties
- Referencing a property
- User views

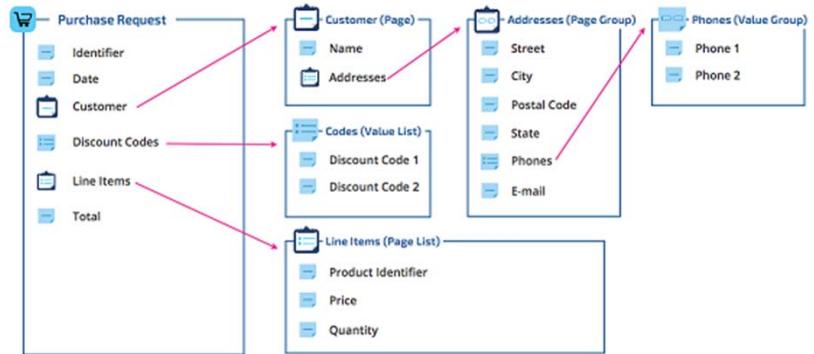
Application developers need to create and modify user interface forms so that they can create a more engaging user experience.

**After this lesson, you should be able to:**

- Define data elements and place them on fields
- Set property values
- Configure calculations based on case data

# Data Elements in Pega Applications

- The data elements or collection of related data elements in a case type comprise the **data model**.
- The data model defines the **case type data structure**.
- A collection of related elements is called a **data type** or **data object**.



# Properties

In Pega, data elements are called **properties** or **fields**, which can be:

- A **single value** with no intended correlation with any other value.
- A collection of related values called a **data type** or **data object**.

Properties are categorized into property **modes**.

- **Value mode** represents a single piece of information.
- **Page mode** represents a data object that contains related values.

# Value Mode Properties

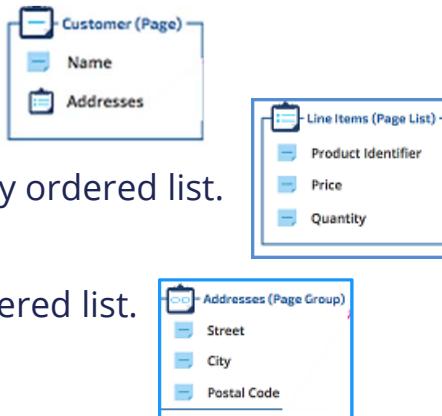
- A **single value** property stores text, numbers, dates, or Boolean values. Single value properties have **property types**.
- A **value group** acts as a container for an unordered list of single values.
- A **value list** acts as a container for an ordered list of single values



# Page Mode Properties

Page mode properties establish contextual relationships between single value properties.

- A **page** is a single entity.
- A **page list** is a numerically ordered list.
- A **page group** is an unordered list.



Pega Platform tools provide easy-to-use interfaces that add, update, and remove classes and properties.

**After this lesson, you should be able to:**

- Manage properties in the Data model tab of the Case Designer

# Managing Properties

- Pega provides several tools that help manage properties. They include the following:
  - Data model tab
  - Selecting the field type
  - Property rule form

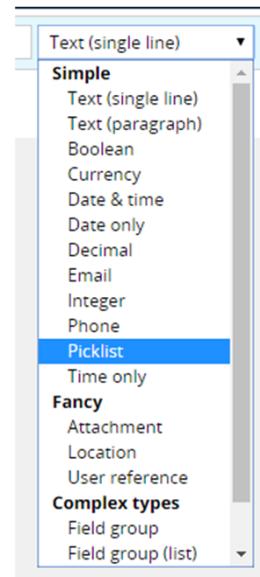
The screenshot shows the Pega Data types interface. On the left, there is a sidebar with categories like Courses, Dental Plan, Medical Plan, Office, and Vision Plan. The 'Office' category is selected. In the center, a modal window titled 'Data types' is open, showing a list of fields: 'Phone type' and 'PhoneType'. A dropdown menu next to 'PhoneType' shows options: 'Text (single line)', 'Simple', 'Text (single line)', 'Text (paragraph)', and 'Boolean'. Below this, another modal window titled 'Edit Property: Marital status [Available]' is open, showing the 'General' tab with settings for 'Property type' (set to 'Text') and 'Data access' (set to 'Manual').

# The Data Model Tab

- You can use the **Data model** tab in the Case Designer to add or remove fields from your case type.
- You can select **Show reusable fields** to display all fields inherited and available in the case type.

# Selecting the Field Type

- When creating a new field, you must specify a type. There are three type options:
  - Simple types** are similar to the property types defined on the property itself. Use a picklist if you need to display a static list of options to the user.
  - Fancy types** allow you to provide the capability to upload an attachment, show a location on a map, or reference a user on the system.
  - Complex types** define page and page list properties. A field group is a page and a field group (list) is a page list.



# The Property Type Rule Form

- The property rule form contains the property definition.
  - **Data Access** section: Configure automatic data access and persistence settings
  - **Display and validation** section: Define how the property should appear on the screen by specifying a UI control
- Pega comes with a set of standard property rules.
- The prefix identifies how you can use a standard property.
  - **px:** Identifies special properties. Your application can read but not write to these properties
  - **py:** You can use these properties in your application.
  - **pz:** Supports internal system processing. Your application can read but not write to these properties.

## LESSON

# Referencing a Property

Based on the data collected, decisions on how to best process and resolve the case are made.

**After this lesson, you should be able to:**

- Reference properties in Pega applications

# Referencing a Property

Refer to a property in Pega by prefixing the property name with a “.” (period or dot).

<i>Value mode properties</i>	.OrderDate	• A single value property named <i>OrderDate</i>
	.Phone(Mobile)	• An entry in a value group property, such as the mobile phone number where Mobile is the group subscript
<i>Page mode properties</i>	.DiscountCode(1)	• The first entry in a value list property, such as one of the discount code
	.Customer	• A page that contains customer information
	.Address(Work)	• An entry in a page group property, such as the work address, type
	.LineItems(3)	• The third page of a page list that contains purchase request line items, type

# Referencing a Property continued

- To refer to a specific property on a page, use the name of the page as a prefix for the property name.
  - Establish an important piece of information about that property — its context.
  - The context of a page — by itself or as part of a page list or page group — acts as a container for the properties it contains.
- If you want the city in the work address, specify **.Address(Work).City**.



DEMO

## How to Define Properties

© 2018 Pegasystems Inc.

15

- Adding a field to a case or data type
- Updating a field in the case or data type
- Remove a field from the case or data type

Pega application users perform tasks by entering information in views. In Pega, you can create views while developing case life cycles.

**After this lesson, you should be able to:**

- Describe the purpose of a user view
- Configure a user view and its data elements while creating a case life cycle

# User Views

- As a case goes through a process, users perform a variety of tasks
- To perform tasks, users enter or review information in user forms. Pega's term for a user form is a user view.

# Views for Specific Tasks

- Views are designed so that users can complete tasks. In order to complete the tasks, users enter information in fields .
- The system stores the values entered by users as data. The application uses this data to process a case.

# Identify the User Tasks

- Create a view in which users enter the specified information.
- Before you create a view, ask yourself the following questions:
  - What fields do users need to see?
  - How will users enter values in those fields?
  - Can users modify the field values or only read the values?



## How to Configure User Views

© 2018 Pegasystems Inc.

20

- Configure a user view using App Studio
- Add default fields to the user view
- Add new fields to the user view
- Save and verify your work
- Designing a picklist



## MODULE Data in Memory

This module contains the following lessons:

- Data storage in memory
- pyWorkPage
- Viewing clipboard data
- Using and setting property values with the Clipboard

Cases are collections of data. To process and resolve a case, Pega applications capture, manipulate, and present data throughout a business process. While processing a case, this data remains in memory for use by one or more users.

**After this lesson, you should be able to:**

- Explain how data is stored in memory for use in Pega applications
- Explain the relation between data elements and pages
- Explain how pages are structured in the Clipboard

# Data Elements and Pages

Each **data element** in a Pega application is a pairing of two pieces of information: the name of the data element, and the value assigned to the data element.

Each data element is stored in memory on a page. A **page** is a structure for organizing data elements in an application.

- Some pages are created *by the system* to track user or session data.
- Other pages are created *by system architects* to describe a data object, such as a hotel reservation or a customer.

# Pega Clipboard

The Clipboard is:

- A portion of memory on the server reserved by Pega for the data generated by applications.
- A collection of all of the pages used to track name-value pairs or data elements.
- A location where pages can be added to or removed from memory as needed to track case or session data.





## GROUP DISCUSSION

© 2018 Pegasystems Inc.

5

How is information, such as the customer's date of birth, stored in memory for use in a Pega application?

When you debug case behavior, you often need to view the case data that is in memory on the clipboard. Viewing the data on the clipboard can help you identify the cause of the issue.

**After this lesson, you should be able to:**

- Describe the relationship between pyWorkPage and case data
- Explain the relationship between pyWorkPage and PyWorkCover
- Explain the importance of class information when referencing data on pyWorkPage

# Structuring Data in Memory

- The **pyWorkPage** is a specific page on the clipboard that stores all the data generated while creating and processing a case.
- An **embedded page** is a page within pyWorkPage that stores data describing a data type.
- The **pyWorkCover** is a separate page that contains the case data for a parent case when the associated child case is processed.

# Referencing pyWorkPage in Rules

Each page on the clipboard is an instance of a specific class. To ensure that the report obtains the correct information whenever you reference pyWorkPage, you need to specify the class of pyWorkPage.

The screenshot shows two panels. The top panel is a table titled 'Page name' and 'Class'. It contains one row with 'pyWorkPage' in the 'Page name' column and 'MyCo-PA-Work-Quote' in the 'Class' column. There are icons for edit and delete at the end of the row. A '+' button is located below the table. The bottom panel is titled 'Edit filters' and contains a section 'Filter conditions to apply' with the value 'A'. Below this is a table with columns 'Condition', 'Caption', 'Column source', 'Relationship', and 'Value'. The first row has 'A' in 'Condition', 'Caption' empty, '.ID' in 'Column source', 'Is equal' in 'Relationship', and 'pyWorkPage.Customer.ID' in 'Value'. There are dropdown arrows and a delete icon between the columns.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

9

What is the purpose of an embedded page within pyWorkPage?

Why is it important to include class information when you reference data on pyWorkPage?

To view data that is in memory, you use the Clipboard tool. The Clipboard tool organizes and presents all the pages on the clipboard. When you select a page, the Clipboard tool lists all the properties on each page and the value of those properties.

**After this lesson, you should be able to:**

- Navigate around the Clipboard tool
- Explain how the Clipboard tool organizes data in memory
- Distinguish the four categories of Clipboard pages

# Clipboard Tool

Use the header (1) to select the **thread** to view. The left pane (2) lists each page defined on the clipboard for the selected thread. The right pane (3) lists all of the properties defined on the selected page, and their values.

The screenshot shows the Pega Clipboard tool interface. At the top, there's a header bar with the title "Pega Clipboard HHR5QMHQUKFUK6KWEQD74KG711G23LBU". Below the header, the "Thread" dropdown is set to "New" (marked with a red circle 1). The left pane (marked with a red circle 2) displays a tree view of pages under the "New" thread, including "D\_pzRBShortcuts", "D\_pzReportBrowser7", "D\_pzReportBrowserExpress", "D\_RAApplicationClassesSimple", "D\_SKinList", "Declare\_AppExplorerData", "Declare\_CaseTree", "Declare\_pyDisplay", "Declare\_pzRecentsCache", "Declare\_RuleTypeMenu", "myParamPage", "newAssignPage", "pyActionInfo", "pyBranchListResultsPage", "pyDataAccessOptions", "pyNavigation1", "pyPortal", "pyReportParameters\_temp", "pyRuleSetVersionContentPage", and "pyWorkPage (Work-ProjectManagement-New)" (which is selected and highlighted in blue). The right pane (marked with a red circle 3) shows a table of properties for the selected "pyWorkPage" page. The table has two columns: "Property" and "Value". Some properties listed include: pxApplication (Reservelt), pxApplicationVersion (01.01.01), pxCreateDateTime (20180516T165802.410 GMT), pxCreateOperator (Author.Reservelt), pxCreateOpName (Author.Reservelt), pxCreateSystemID (pega), pxObjClass (Work-ProjectManagement-New), pxUpdateDateTime (20180516T165802.463 GMT), pxUrgencyWork (10), pxUrgencyWorkClass (10), pyAgeFromDate (20180516T165802.411 GMT), pyCancelLabel (Cancel), pyClassListType (Implementation), pyConfirmationNote (Thank you for your input), pyElapsedStatusNew (0), pyElapsedStatusOpen (0), pyElapsedStatusPending (0), pyFlowKey (RULE-OBJ-FLOW WORK-PROJECTMANAGEMENT-NEW NEWMODALFLOW #20150630T135306.758 GMT), pyFlowName (NewModalFlow), pyFocusKey (pyStreamName), pyFocusKeyRemoveSpaces (true), pyFolderType (pyDefault), pyLabel (New Instance of a Rule), pyLabelOld (New Instance of a Rule), pyNextEmailThreadId (1), and pyNextLabel (Next >>).

Property	Value
pxApplication	Reservelt
pxApplicationVersion	01.01.01
pxCreateDateTime	20180516T165802.410 GMT
pxCreateOperator	Author.Reservelt
pxCreateOpName	Author.Reservelt
pxCreateSystemID	pega
pxObjClass	Work-ProjectManagement-New
pxUpdateDateTime	20180516T165802.463 GMT
pxUrgencyWork	10
pxUrgencyWorkClass	10
pyAgeFromDate	20180516T165802.411 GMT
pyCancelLabel	Cancel
pyClassListType	Implementation
pyConfirmationNote	Thank you for your input
pyElapsedStatusNew	0
pyElapsedStatusOpen	0
pyElapsedStatusPending	0
pyFlowKey	RULE-OBJ-FLOW WORK-PROJECTMANAGEMENT-NEW NEWMODALFLOW #20150630T135306.758 GMT
pyFlowName	NewModalFlow
pyFocusKey	pyStreamName
pyFocusKeyRemoveSpaces	true
pyFolderType	pyDefault
pyLabel	New Instance of a Rule
pyLabelOld	New Instance of a Rule
pyNextEmailThreadId	1
pyNextLabel	Next >>

© 2018 Pegasystems Inc.

11

# Four Categories of Clipboard Pages

- **User Pages** category contains pages created due to user action, either directly or indirectly.
- **Data Pages** category contains read-only data pages defined by data page rules.
- **Linked Property Pages** category contains read-only pages created by linked properties, which contain information from data objects referenced by a linked property.
- **System Pages** category contains pages that describe the current user session, such as the active user and the active application.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

13

While testing case behavior for an online shopping application, you want to confirm that the application properly generates a list of the customer's previous orders when querying the company's order management system. In which category of clipboard pages would you expect to find the page that contains this list?

You use the Clipboard tool to view the properties and values entered in a specific view. This makes the Clipboard an ideal tool for debugging.

**After this lesson, you should be able to:**

- Use the Clipboard tool to review case data in memory
- Use the Clipboard tool to set values for case data



## DEMO

### Using and Setting Property Values with the Clipboard

© 2018 Pegasystems Inc.

15

How to use the Clipboard tool for debugging a child case by adding property values normally obtained from the parent case



## MODULE Manipulating Case Data

This module contains the following lessons:

- Data transforms
- Setting values with data transforms
- Setting default property values
- Data transforms and superclassing

## LESSON Data Transforms

As you process a case, you may need to copy or manipulate data. For example, you collect an individual's first name and last name, but want to combine them into a full name.

**After this lesson, you should be able to:**

- Use data transforms in an application

# Use of Data Transforms in an Application

The purpose of a **Data transforms** is self-explanatory: it transforms data in the application. You can use data transforms in several ways:

- Copy, update, or initialize data
- Act on individual properties or entire pages and iterate over page lists

The screenshot shows a mobile application interface with two address entry forms. The top form is labeled "Shipping Address" and contains fields for Address 1 (#321), Address 2 (Main St), State (Cambridge), Country (United States), and Pin-code. The bottom form is labeled "Billing Address" and contains similar fields. A checkbox labeled "Same as shipping address" is checked, and an arrow points from the "Address 1" field in the Billing form to the "Address 1" field in the Shipping form, indicating that the Billing address is being set to the same value as the Shipping address.

In general, data transformation involves mapping data from a source to a target as well as performing transformations on that data required to achieve the intended mapped results.

**After this lesson, you should be able to:**

- Configure a data transform
- How to reference a property
- Adding a data transform to a process

# Configuring a Data Transform

- Configuring a data transform requires specifying an action and then entering the values for the transform.
- Actions are the individual operations specified in each row on the **Definition** tab of a data transform.

The screenshot shows the 'Edit Data Transform' interface for a record named 'pyDefault [Available]'. The top bar displays the record name, its status as 'Available', and its ID 'pyDefault'. Below the top bar, a message indicates 'This record has 1 unreviewed warning (review/edit)'. The main area features a table with two columns: 'Action' and 'Target'. A single row is present, showing '1' in the Action column and 'Apply Data Transform' in the Target column. The Target dropdown menu is open, showing the option 'pySetFieldDefaults'. The table has a header row with the column names 'Action' and 'Target'. There are also icons for copy, paste, and delete at the bottom right of the table.

# How to Reference a Property

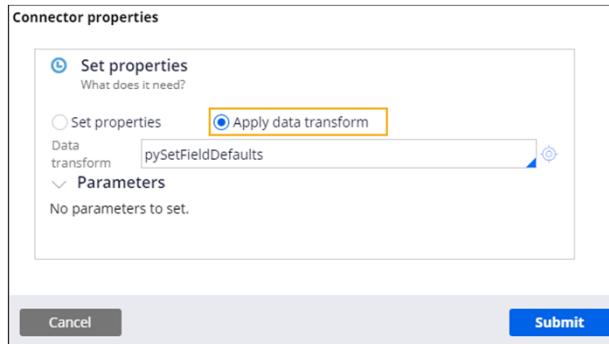
There are two property modes: **value** and **page**.

- Page mode properties act as a container for value mode properties.
- You refer to a property in by prefixing the property name with a period (or dot, ".").
- To refer to a specific property on the page, use the name of the page as a prefix for the property name.

Definition	Parameters	Pages & Classes	Specifications	History
	Action	Target	Relation	Source
# • 1	When	.BillingSameAsShipping		
	• 1.1	Set .Customer.Address(Billing)	equal to	.Customer.Address(Shipping)
# • 2	Otherwise			
	▼ • 2.1	Update Page .Customer.Address(Billing)		
	• 2.1.1	Set .AddressLine1	equal to	“”
	• 2.1.2	Set .AddressLine2	equal to	“”

# Adding a Data Transform to a Process

Most data transforms occur between assignments. Double-click the connector between the assignments to open the connector properties panel. In the Connector properties panel, under Set properties, select **Apply Data Transform**.





DEMO

## How to Set Values with Data Transforms

© 2018 Pegasystems Inc.

8



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

9

What is an action in a data transform?

Briefly describe the two types of property modes.

Where do most data transforms occur in a process flow?

When a user creates a case, they may want to set default values for some properties. For example, in an insurance claim case, you could set the date of loss to today's date.

**After this lesson, you should be able to:**

- Set property values using *pyDefault* and *pySetFieldDefaults*

# Setting Default Values for Properties

- **pyDefault** data transform is invoked when a case is created. It is used to set default values for cases.
- **pySetFieldDefaults** data transform is used to initialize default field values.
- *pyDefault* uses the **Apply Data Transform** action to call *pySetFieldDefaults*.

The left side shows the navigation tree under 'Onboarding' with 'Data Model' expanded, showing 'Data Transform' with 'pyDefault' and 'pySetFieldDefaults' listed. The right side shows the 'Edit Data Transform: pyDefault [Available]' screen. The top bar includes the application name 'TGB-HRApps-Work-Onboarding', ID 'pyDefault', and version 'HRApps.01-01-01'. A message indicates 'This record has 1 unreviewed warning (review/edit)'. The main area has tabs for 'Definition', 'Parameters', 'Pages & Classes', 'Test cases', 'Specifications', and 'History'. Under 'Definition', there's a table with one row. The 'Action' column contains 'Apply Data Transform' with a dropdown menu showing 'pySetFieldDefaults'. The 'Target' column is empty. There are also icons for copy, delete, and refresh.



DEMO

## How to Set Default Property Values



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

13

You are using *pyDefault* to set the manager property to Sharon Smith. What do you use as your Target and Source values?

What action is defined in *pyDefault* to ensure that *pySetFieldDefaults* is invoked when a case is created?

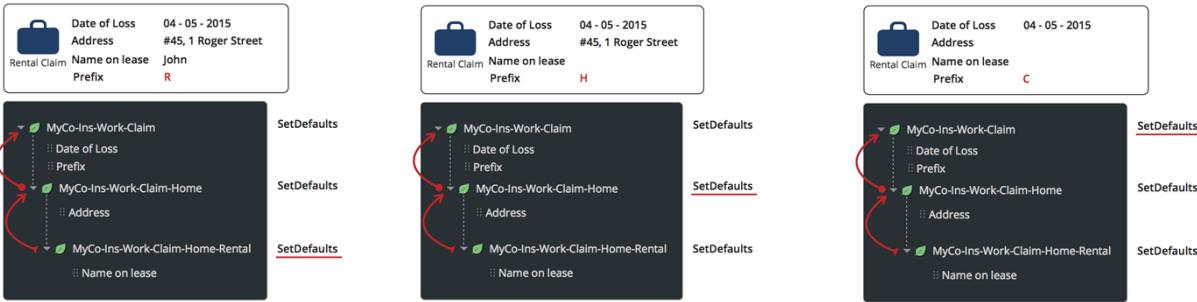
You can have a class called Claim with a subclass called Home. The subclass Home in turn has a subclass called Rental with data transforms at each level that sets default values.

**After this lesson, you should be able to:**

- Explain how data transform superclassing works

# Configuring Superclassing for Data Transforms

- You can combine several data transforms using the superclass feature to set values at multiple levels of the class hierarchy. Superclassing calls data transforms with the same name at multiple levels of the class hierarchy.



The Prefix value changes from R to H to C as each data transform is invoked at different levels of the class hierarchy



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

16

What data transform feature do you use when you want to set a different default value for the same property within a hierarchy of subclasses?



This module contains the following lessons:

- Declarative processing
- Declare expressions
- Setting a property value with a declare expression
- Forward and backward chaining in declarative networks

Application developers need to identify the differences between declarative processing and procedural processing.

**After this lesson, you should be able to:**

- Describe the declarative processing model
- Describe the procedural processing model

# Declarative and Procedural Processing

- **Declarative processing** instructs the system to monitor the application to determine when a trigger event occurs. Declarative rules define a trigger event and resulting action.
- **Procedural processing** depends on rules to instruct the application when to look for a trigger event. Rules such as data transforms, activities, or user interface (UI) rules perform updates based on a predefined trigger.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

4

What is the primary benefit of using declarative processing?

Which technique should you use if you want to update values when the user submits a form?

Which technique should you use if you want the total price to update immediately when a quantity is changed?

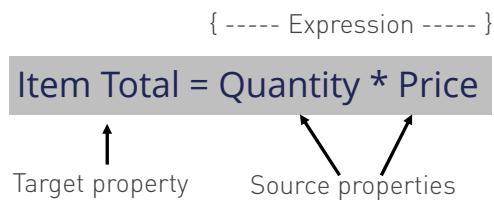
Application developers need to identify the components of a declare expression and how they relate to declare networks.

**After this lesson, you should be able to:**

- Calculate a property value with a declare expression

# Declarative Expressions

- **Declare expressions** automatically calculate property values and are comprised of an **expression** and a **target property**.
- The expression determines the calculated value and updates the target property. The expression references one or more **source properties**.



# Declarative Networks

- A **declarative network** is a set of interdependent declare expressions.
  - For a list of declarative networks in an application, select **Designer Studio > Process & Rules > Business Rules**.
- A declare expression can use the target property from another declare expression as a source property.



DEMO

## How to Set a Property Value with a Declare Expression

© 2018 Pegasystems Inc.

8



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

9

What are the three components in a declare expression?

A declare expression in a network can use which property as a source property?

Where are the source values entered when creating declare expressions in Case Designer and Dev Studio?

## LESSON

# Forward and Backward Chaining in Declarative Networks

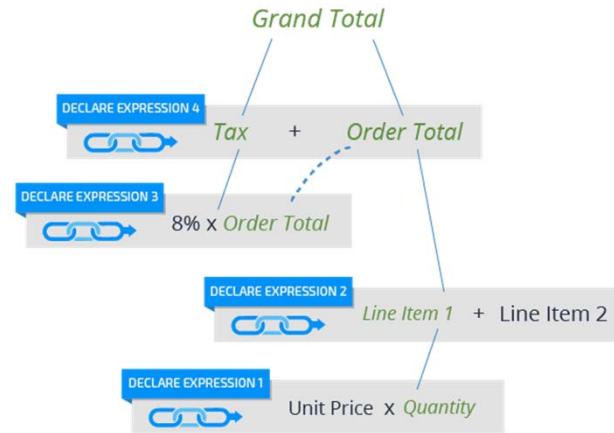
Application developers need to identify how the chaining method affects declare expression execution.

**After this lesson, you should be able to:**

- Explain how forward chaining works to update property values
- Explain how backward chaining works to update property values

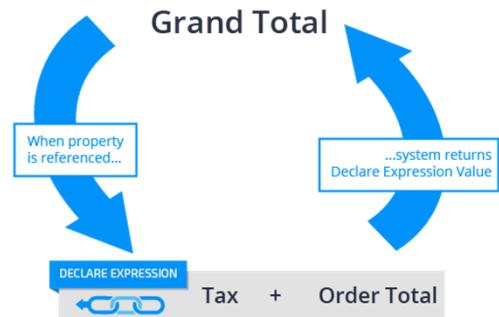
# Forward Chaining

- **Forward chaining** in a declare expression pushes updates to the target value
- Updates the target property value when a source property value changes



# Backward Chaining

- **Backward chaining** pulls values from the source property or properties.
- The target property value does not automatically update when a source property changes.
- The target property is updated when the application references the property by name.



# Backward Chaining Options

Option	Description
When used, if property is missing	Compute when the target property is not present on the clipboard.
When used, if no value is present	Compute when the value is null, blank, zero, or does not yet appear on the page. Later requests for the target property do not cause the declare expression to run.
Whenever used	Compute even when the property has a value.

# Chaining and Performance

- Consider how a property is used and the frequency with which source values change.
- Forward chaining can slow system performance if an expression uses many source properties that change frequently.
- Backward chaining may avoid unnecessary updates and improve response time.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

15

When does an expression using backward chaining update its target property?

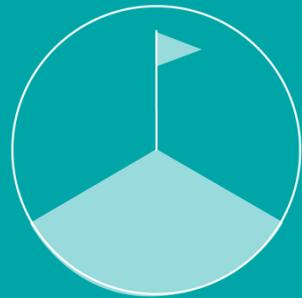
When can forward chaining have a negative impact on performance?



DEMO

## How to Define and Test a Declare Expression





# CAPSTONE EXERCISE

## Designing a Data Model

© 2018 Pegasystems Inc.

## MODULE Validating Case Data

This module contains the following lessons:

- Methods of data validation
- Validating case data with controls
- Validating case data with validate rules
- Validating a flow action with a validate rule
- Using edit validate rules

When you design a view, and add all the fields and controls that the specification requires, you must also consider how to ensure that the data values entered by users are valid. Valid data is required so that the system can process the information without error.

**After this lesson, you should be able to:**

- Describe the methods of data validation
- Describe important design requirements
- Explain how to use properties and controls to validate data
- Describe how to use validation rules to validate user input

# Important Design Requirements

The data must:

- Be the right type.
- Fit the business logic.
- Be restricted to possible values.

Pega also provides property types, controls, and rules to support validation requirements.

# Properties and Controls

- **Properties**

Single value properties have property types such as date, decimal, integer, text, or true/false. Selecting the appropriate property type ensures that users enter a valid value.

- **Controls**

Controls are another way you restrict users from entering or selecting invalid values on a form.

# Validation Rules

- You use validation rules when you cannot predict or control the value users enter in a form.
- There are two types of validation rules: **validate** and **edit validate**.
  - **Validate rules** to compare a property against a condition when the user submits a form.
  - **Edit validate rules** to test single value, value list, and value group properties for patterns.
- By default, Pega Platform performs client-side validation.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

6

1. You have added fields for entering the name and address of a loan applicant. What validation methods would you use?
2. Which validation method would be appropriate for checking that a user enters a date for scheduling a home inspection that is in the future?

Controls used on views provide the most common approach to validation. The three basic ways you can use controls for validation are control types, required fields, and editable settings.

**After this lesson, you should be able to:**

- Explain how to use control types to validate user data
- Illustrate how to configure a property as a required field
- Explain how to configure editable settings

# Control Types

Use case	Control type	How the control helps validation
Users must enter a date that includes day, month, and year.	Calendar	Selecting a date from a calendar icon helps ensure that users enter a date in a valid format.
Users must select one of three possible loan types. Users must see all types on the form.	Radio buttons	Restrict choices to a set of valid values and allows users to select only one value. You can use radio buttons when only a small number of options (for example, fewer than five) is available.
Users must select one of 10 types of office chairs from a list. The options do not need to be displayed on the form.	Dropdown	Restricts valid values to the ones that appear in the list. A drop-down list presents the options only when users click the control. This helps reduce the clutter on the form. Unlike radio buttons, you can configure the drop-down control so that users can select multiple values.
Users must select the country in which they reside from a list. Users can enter text in the control to help find the right country.	Autocomplete	When users enter one or more values in the control, the control filters the available options. This helps users find an option in a list if there is a large number (for example, more than 20) of possible options.
Users select an option to purchase extra travel insurance.	Checkbox	User can select the check box or leave it blank. This ensures that a true/false property is either true (selected) or false (unselected).

# Specifying Control Types

- Most of the fields you add in the **Data Model** tab in either Pega Express or the Case Designer are associated with default control types.
- The control type depends upon the field type you select.

Field type	Control type
Date only Date & Time	Calendar
Boolean	Check box
Picklist	Radio buttons or a drop-down (you choose the control type when you add the property type)
Text (paragraph)	Text area

- Pega standard controls are designed to display in native format on the client.

# Required Field

- Configuring a control as a required field ensures that users enter a value. If there is no value, users receive an error when they try to submit a form.

Omitting a date of birth prompts the user with an error message when the user attempts to submit the form.

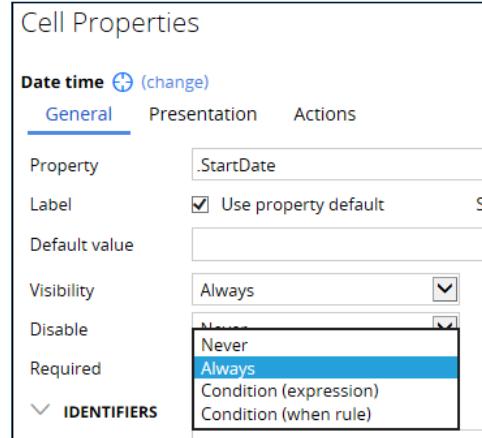
A screenshot of a web browser window showing a form submission process. The form has a single field labeled "Date of Birth \*". Below the field is a red error message: "Must enter date of birth". A blue "SUBMIT" button is at the bottom of the form.

The error message does not appear if there is a date in the field.

A screenshot of a web browser window showing a form submission process. The "Date of Birth \*" field now contains the value "03/31/2002", which is highlighted with a green checkmark icon. The red error message from the previous screenshot is no longer present. A blue "SUBMIT" button is at the bottom of the form.

# Configuring a Required Field

- In a view, select **Required** in the **Options** drop-down for a required field.
- In a section, open the field's Properties panel. In the **Required** drop-down list, select Always so that users must enter a value under any condition.



# Editable Settings

- You can use editable settings on controls to restrict the input values to valid formats.
- The settings are specific to the control type.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

13

1. How would you configure a field in which a user selects one of four possible shipping methods?
2. When would you use the required fields validation approach?
3. How would you ensure that a user always enters 20 characters in a field?

## LESSON

# Validating Case Data with Validate Rules

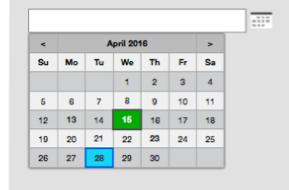
Property types and the associated controls — for example, a date type property that uses a date picker control. However, these types of validation methods cannot ensure the input provided by end users conforms to business policies.

### **After this lesson, you should be able to:**

- Describe how to validate case data using validate rules

# Validating Case Data with Validate Rules

- Property types and the associated controls cannot ensure the input provided by end users conforms to business policies.

Validate Start Date is Before Current Date		Validate Start Date is After Current Date	
<b>Job History</b>		<b>New Hire</b>	
			
 Today	 Start Date	 Today	 Start Date

For example, a job history form requires the start date of employees which must be past dates. A new hire form requires the start date of employees to be entered as future dates. Validate rules can ensure the requirements are met on both forms.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

16

Validate rules enable you to use a single property when \_\_\_\_\_.

**After this lesson, you should be able to:**

- Validate data in App Studio
- Validate data in Dev Studio
- Create the validate rule
- Associate the validate rule with a flow action

# Validating a Flow Action with a Validate Rule

- Validating an input field with a validate rule is a two-step process.
  - Create and define a validate rule.
  - Update the flow action properties panel to specify the validation rule you want to use.



DEMO

## Validate Data in App Studio

# Validate Data in Dev Studio

- In Dev Studio, you can create validate rules that model more complex validation conditions, such as a condition that compares the results of two fields or a condition that requires a function to determine the comparison value.
- In addition, you can create a validate rule in Dev Studio to provide an entry condition for a stage in the case workflow.



DEMO

## Create the Validate Rule



DEMO

## Associating the Validate Rule with a Flow Action



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

23

Why must a validation condition that tests whether a date is in the future or the past be configured in Dev Studio?

The logic in edit validate rules is written as a Java procedure. Pega provides a large set of the standard edit validate rules so you do not have to create your own rules.

**After this lesson, you should be able to:**

- Describe how edit validate rules can validate user data
- Explain how to use edit validate rules

# When to Use Edit Validate Rules

- **Edit validate rules** validate character patterns.
- For example, you can use an edit validate rule to ensure that the field contains the @ symbol in an email address. If it does not, an error message appears when the form is submitted.

The figure consists of two side-by-side screenshots of a computer monitor displaying a web form. Both screenshots show a single input field labeled 'e-mail\*' containing the text 'JohnSmith\_Pega.com'. In the left screenshot, a red error message 'Enter a valid email address' is displayed below the input field, and the 'SUBMIT' button is grayed out. In the right screenshot, the input field now contains the correct email address 'JohnSmith@Pega.com', which is followed by a green checkmark icon, indicating successful validation. The 'SUBMIT' button is now active and blue.

# Using Edit Validate Rules

- You can associate a single edit validate rule with a property.
- You can also reference edit validate rules in a validate rule.
- This approach enables you to apply multiple edit rules to a single property.

- To associate an edit validate rule with a property, open the property rule. The property must be a single value, value list, or value group. Open the Advanced tab and select an edit validate rule in the **Use validate** field.
- To use an edit validate rule with a validate rule, open the validate rule. In the **Select a function** field, select the function *Validation of [Property Name] using [Edit Validate Name]*. In the **Validation of** field, enter the property you want to validate. Then in the **using** field, select an edit validate rule.



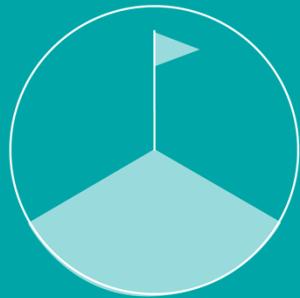
## GROUP DISCUSSION

© 2018 Pegasystems Inc.

27

You have added a field for entering a U.S. phone number. Do you use a integer data type or an edit validate rule to validate that the phone number is in the correct format?





## CAPSTONE EXERCISE

### Configuring Data Validation

© 2018 Pegasystems Inc.

## MODULE Using the Integration Designer

This module contains the following lesson:

- Visualizing data with the Integration Designer

## LESSON

# Visualizing data with the Integration Designer

As an application developer you can view and navigate the application data model, how the application uses it, and the system of records that the application connects to.

### **After this lesson, you should be able to:**

- Navigate to the Integration Landscape view
- Navigate to the Integration Designer landing page
- Navigate directly to a data type
- View relationships between data types and systems of record



DEMO

## How to Visualize Data with the Integration Designer

© 2018 Pegasystems Inc.

3



## MODULE Creating Data Types

This module contains the following lessons:

- Application data processing with data types
- Create a locally sourced data type
- Configure a data source for an existing data type
- How to create an externally sourced data type

## LESSON

# Application Data Processing with Data Types

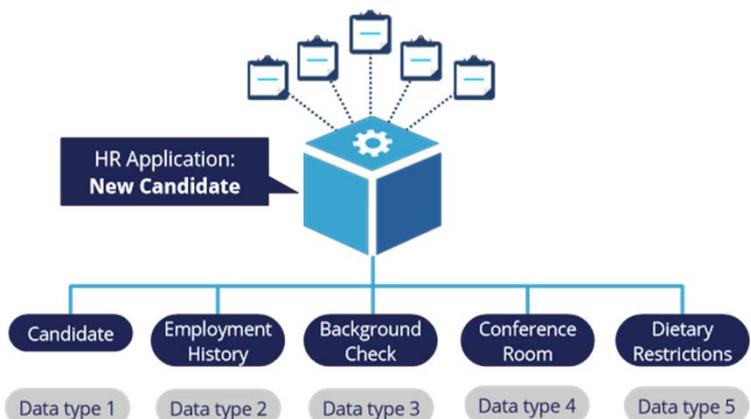
Application developers need to be proficient at using data types to effectively manage the data requirements for a Pega application.

### **After this lesson, you should be able to:**

- Indicate when a data type is required in an application
- Identify local, system of record, and reference data use cases for data types
- Differentiate Pega data types from external system data types
- View relationships between data types and systems of record

# Data Types

- Pega applications collect, consume, and process data that is important to a case. **Data types** uniquely define and hold data for your application.
- A Pega application uses many different data types to process a case.



# Data Type Sourcing

- **Pega system of record** – A Pega system of record can locally source data types.
  - **Example:** In the New Candidate application, HR can select a conference room in which to conduct the interview. The Pega system of record locally sources the **Conference Room** data type and the Pega system of record would have a list of the conference rooms.
- **External system of record** – External systems can source data types.
  - **Example:** A **Candidate** data type stores basic identifying information about candidates. An external system of record, (external vendor tool that adds and tracks candidates), sources the Candidate data type.

# Data Type Sourcing (continued)

- **No system of record** – Data types can obtain data entered or transformed during application processing and not associate with a system of record.
  - **Example:** During interview scheduling, you capture candidate information about any dietary restrictions. This helps HR provide an appropriate lunch for the candidate during the interview break. The **Dietary Restrictions** data type is used for case processing but is not pushed to a system of record.

# Data Model Type and Structure

- The collection of case types and data types in your application define your data model holistically.
- Each data type defines a specific data structure. The data structure consists of one or more fields, also referred to as properties.
- Data types can reference other data types.

# Using Data Types

- Use standard Pega data types when possible such as ***Data-Address-Postal*** and ***Data-Address-Email***.
- Extend an existing data type if it only partly meets your needs such as extending the existing ***Pega Party-Person*** data type to create the ***Pega-Party-Person-Employee*** data type.
- If no suitable data type already exists, then create a new one.

# How Data Types Are Used

- Data types can be used for reference data.
- Data types can be added to case types and persisted with the case.

## Reference data examples:

**Ex 1:** You have an airport code drop-down list. The drop-down list is populated by an **Airport Code** data type. An external system of record sources the Airport Code data type. Only the user-selected airport code would be added to the case type.

**Ex 2:** You have an airline reservation case type where users are shown various flight options. There is a data type for **flight selections** and for **potential flights**. The data is only for display and is not stored in the case. Only user-selected flights are added to the case.

## Case types persisted with the case examples:

A **Customer** data type is added to a case type to define its data model. The Customer data type is externally sourced. Your application reads the external system of record for a list of customers. For new customers, data entered into your application is then pushed to the system of record.



DEMO

## How to Create a Locally Sourced Data Type



DEMO

## How to Configure a Data Source for an Existing Data Type



DEMO

## How to Create an Externally Sourced Data Type



## MODULE Managing Data Pages and Views

This module contains the following lessons:

- Data views and data pages
- Accessing on-demand data with a data page
- Configuring data page sourcing options
- Saving data with a data page

Application developers need to understand the capabilities of data pages to access and persist data on demand from various sources.

**After this lesson, you should be able to:**

- Define the purpose of a data page
- Differentiate thread, requestor, and node-level scope data pages
- Define scope, structure, edit mode and object type
- Configure a refresh strategy
- Explain the purpose and use case of edit modes

# Understanding Data Pages

- Use **data pages** to retrieve data for your application, regardless of the source.
- Data pages cache data on demand to a clipboard page and define who can access the data page.
- In Dev Studio, you create **data views**, also known as **data pages**. In
- In App Studio, you can view a list of **data views** associated with a data type.

# Defining a Data Page

- **Data page scope** – captures how widely data is visible in an application. There are three levels of a scope: **thread, requestor and node**.
- **Data page source** – any source of data that an application uses, such as a connector, report definition, or lookup. Data sources are referenced in data pages.
- **Structure of a data page** – uses a single page to load a single record. You use a list if you want to load multiple records.

## Defining a Data Page (continued)

- **Refresh strategy** – defines when the data is stale and needs to be reloaded. Data pages are created and updated on demand. The page is never reloaded until it is referenced.
- **Data page object type** – The object type identifies the information the data page will capture.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

6

Consider the following set of use cases (use as a basis for starting the discussion):

1. Obtain data from an external source
2. Return a list of data objects mapped in the application
3. Return a specific data object mapped in the application
4. Situation where no other options are suitable

Answers

1. Connector
2. Report definition
3. Lookup
4. Activity

# Data Sourcing

- Data source abstraction allows developers to use data page content without knowing how the data is sourced.
- A data page can conditionally source from multiple sources.
- When sourcing with parameters, you specify parameters the system can use when referencing a particular data page.

# Refresh Strategy Configuration

- A data page remains on the clipboard in memory to serve requests without reloading the data.
- A data page is configured to reload when older than a designated time frame.
- It is not reloaded until the next request for the page occurs following the configured time.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

9

Refresh a list of currency exchange rates to ensure that data is always current - Reload once per interaction

Refresh a list of vehicle makes and models that is more than one month old - Reload if older than

Refresh a list of asset prices once the trading day ends - Do not reload when [condition]

# Defining an Edit Mode

- **read only** – Used for data pages that should not be modified. The data page displays in the Data Page list on the clipboard.
- **editable** – Allows the data page to be modified. The data page is displayed in the user page list on the clipboard.
- **savable** – Provides the data page by saving through a database source or an activity. Savable data pages are referenced in save data page locations. Selecting this mode opens the data save options sections so that you can select a save plan option.



DEMO

## How to Access On-Demand Data with a Data Page



DEMO

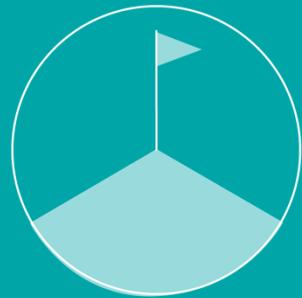
## How to Configure Data Page Sourcing Options



DEMO

## How to Save Data with a Data Page





## CAPSTONE EXERCISE

### Configuring Reference Data

© 2018 Pegasystems Inc.

This module contains the following lessons:

- Section rules
- Guidelines for designing user views
- How to configure a section
- How to configure responsive UI behavior

Good user interface (UI) design considers the impact of device size and how users interact with their devices. Application developers use sections and layouts to create user views.

**After this lesson, you should be able to:**

- Define the purpose of a section within Pega
- Differentiate between a dynamic layout and a repeating layout
- Configure section layouts using design templates

# Section Rules

Section rules are created automatically when you create user views in the case life cycle.

- Users interact with an application and perform tasks through user forms.
- In Pega, you build user forms with **sections**. Sections group information and functionality by context and purpose.
- Inside a section, you organize UI elements with **layouts**. Layouts contain rows and columns, defining a set of **cells**.

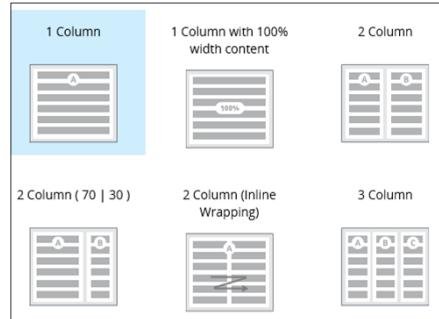
# Sections and Dynamic Layouts

Sections use dynamic layouts that arrange items in a flexible form, automatically adjusting to screen size.

- **Dynamic layout** – Organizes a single set of fields in a general purpose layout.
- **Repeating layout** – Organizes lists, tables, and other repeating structures.  
Repeating layouts reference a page list or page group.

# Design templates

- Use **design templates** to configure section layouts. Templates:
  - Provide a set of commonly used UI designs
  - Make it easy to add UI elements to a section without manual layout configuration
  - Often use dynamic layouts
  - Consist of one or more layouts and embedded sections
  - Enable you to logically group elements and reuse them in other layouts





## GROUP DISCUSSION

© 2018 Pegasystems Inc.

6

What is the relationship between a section and a user view?

Describe the purpose of dynamic layouts

What is one advantage of design templates?

The user interface (UI) is the user's view of the application. Effective UI is intuitive and meets the needs of the user.

**After this lesson, you should be able to:**

- List the benefits of using an intent driven UI
- Articulate the benefit of using a model driven UI

# Design a Model and Intent-Driven UI

- Provide users with information when they need it.
- Align the user view with the case life cycle to ensure the user interface is model-driven.
- A model-driven approach results in faster application development, and a contextually sensitive UI.
- A model-driven approach enables you to quickly modify the UI when business rules change.

# Guidelines for a Positive User Experience

- Keep the UI simple and obvious
  - The best interfaces are almost invisible to the user.
  - Minimize data elements to an optimal set that is critical to the task. Focus on what users are trying to accomplish.
  - Be aware of where users are in the life cycle and the next best action.
  - Users should not have to guess how to proceed while looking at a view.
- Use common UI labels and elements
  - Users are more comfortable and complete tasks faster when views are consistent within an application.
  - Create patterns in language, layout, and design to help users complete their tasks.
  - Design consistency enables users to recognize similar tasks in other application views.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

10

## LESSON Configuring a Section

The section is the fundamental UI rule in Pega applications. When you create user views in the case life cycle, Pega automatically creates a section rule. Application developers format sections with the correct controls and layouts.

### After this lesson, you should be able to:

- Define label, class, ruleset and version
- Arrange a dynamic and repeating layout in section
- Add controls to a section

# Configuring Sections

- **Label** – Used to identify the purpose of the section. The label is used to generate an identifier for the record.
- **Class** – The class to which the record is applied. The class determines the availability and re-usability of the section.
- **Ruleset and version** – The ruleset and version that contains the record. Pega defaults to the highest unlocked application ruleset when you create a record.

# Layouts

- A layout is a container that governs the positioning of fields. Section rules generally use two types of layouts: **dynamic layouts** and **repeating layouts**.
- Use **dynamic layouts** to arrange items in a flexible form, automatically adjusting to screen size.
- Use **repeating layouts** displaying a collection of data that belongs to a page list or a page group.

# Controls

- After you create a section, add controls and map the controls to specific data elements.
- For each control, you can configure the presentation and add an action set as appropriate.



DEMO

## How to Configure a Section



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

16

What are the three ways a section can be configured?

Describe the purpose of dynamic layouts

## LESSON

# Configuring Responsive UI Behavior

People expect easy, user-friendly access to applications on all their devices, such as laptops and mobile phones. Application developers will be able to create applications using responsive UI.

### **After this lesson, you should be able to:**

- Design a layout using responsive breakpoints
- Configure a responsive table using application skins

# Responsive UI

- A **Responsive UI** enables a layout to automatically adjust to rendering devices. Elements can move around, resize, or completely disappear based on screen resolution and size.
- Pega applications have system default **responsive breakpoints** that define when changes in behavior on different devices skin rules should occur
- Responsive breakpoint configuration can be found ~~occurs~~-in the application **skin rule**. A skin rule defines presentation formatting instructions for one or more UI forms.

# Configuring a Responsive Table

- Relying on breakpoints alone for tables can result in unimportant information having prominence over more important information.
- For a table, you use responsive behavior to hide less important columns to make room for the columns with vital information.
- In Pega, importance settings are primary, secondary, and other.
  - **Primary** – The unique identifier for the row. Each table must have one primary column.
  - **Secondary** – Important information with a significant effect on usability. Omitting information in a secondary column impacts the ability of the user to complete a task.
  - **Other** – Information with minimal impact on usability



DEMO

## How to Configure a Responsive UI Behavior



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

21

What are responsive breakpoints in Pega application?

Why do tables need responsive configuration in addition to responsive breakpoints?



## MODULE Adding Dynamic Content to User Views

This module contains the following lessons:

- Dynamic UI design
- Dynamic UI design configuration
- Event processing
- Event processing configuration

## LESSON

# Dynamic UI Design

In a dynamic user interface (UI), the UI content changes based on user interactions with the content. Application developers design UI content to provide end users the most efficient platform experience.

### **After this lesson, you should be able to:**

- List the benefits of using a dynamic UI design
- Define the event action model
- List the two types of event action models

# Using a Dynamic UI Design

Using a dynamic UI has many benefits that lead to a more compelling and modern user experience. Creating a dynamic UI provides you with the following benefits:

- Real-time response to end-user behavior
- Robust functionality available for most user interactions
- Reduced visual clutter on the screen
- Fewer full page refreshes, resulting in improved UI responsiveness

## Using a Dynamic UI Design (continued)

- Pega Platform automatically creates a section rule for each user view
- Dynamic behavior is configured in section rules
- Dynamic behavior enables application developers to hide portions of the UI until those portions are needed

# Event-Action Model

When designing a dynamic UI, you use an **event-action model**. Think of event and action as a cause-and-effect pair.

- Two types of events exist: **property-based events** and **user events**.
- **Property-based events** occur either when a data value changes or when a value meets specific criteria.
- **User events** occur when an end user takes some action on the page, such as selecting an option or clicking on a link.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

6

1. How do application developers benefit from dynamic behavior?
2. How would you describe the event-action model?

## LESSON

# Dynamic UI Design Configuration

Pega Platform provides the ability to control how fields are displayed. Application Developers create field attributes for dynamic display.

### **After this lesson, you should be able to:**

- List the benefits of using a dynamic UI design
- Define the event action model
- List the two types of event action models

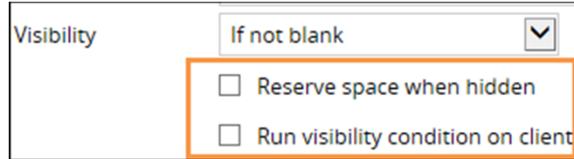
# Configuring Dynamic Content

You can add conditions to UI elements that affect visibility and user interaction

Setting	Definition	Attributes
Always	The UI element is always displayed	Visibility, Disable, Required
Condition (expression)	Uses a Boolean expression to determine visibility, visible when the expression returns true	Visibility, Disable, Required
Condition (when rule)	Uses a when rule to determine visibility, visible if the when rule returns true	Visibility, Disable, Required
If not blank	Visible if the value of that field is not blank	Visibility, Disable, Required
If not zero	Visible if the value of that field is a non-zero number	Visibility, Disable, Required

## Additional Visibility Options

There are two additional options available when you select a visible when condition:



- **Reserve space when hidden:** Keeps the space surrounding a control open. This prevents the UI elements on the screen from repositioning when the visible content is displayed
- **Run visibility condition on client:** When this option is selected, the clipboard page includes all the possible data it can display. The application uses the data on the page to refresh the section based on the visibility condition



DEMO

## Configuring a Visible When Condition on a UI Element



DEMO

## Configuring a Disable Condition on a UI Element



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

12

When do you use a visible when condition?

What is your first consideration when configuring field attributes for dynamic display?

An action set consists of an event, an action, and (optionally) conditions. Application Developers use action sets to configure dynamic UI changes based on the event-action model.

**After this lesson, you should be able to:**

- Define an action set
- Explain event, action and condition
- List six events and six actions

# Action Sets

- **Event:** A trigger performed by users, such as clicking a button, hovering a pointer over a field, or entering a value in a field.
- **Action:** A response performed by the system as a result of the user event.
- **Conditions:** A restriction such as when rules can be applied to an event and action combination.

# Action Sets Examples

The following table shows a few examples of action sets.

Event	Action
Click a control such as a button, link or icon	Opens a new window
Double-click a row in the grid	Opens the row in edit mode
Right-click the entire grid	Shows a menu
Press the Escape key on the keyboard	Closes the assignment and returns to the home page
Select a value from the country drop-down	Updates the list of states/provinces
Select a check box	Unmasks the password
Enter a value in the quantity field	Calculates the total



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

16

What is the purpose of an action set?

## LESSON

# How to Configure Event Processing

Application developers configure an action set for common UI controls, such as fields, buttons, and drop boxes.

### **After this lesson, you should be able to:**

- Configure event processing
- Define action set
- Configure an action set



DEMO

## How to Configure Event Processing



DEMO

## Configuring Event Processing



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

20

What are the main considerations when preparing to configure an action set?

How do you configure an action set to occur when the control is in read only mode?



## MODULE Styling an Application

This module contains the following lessons:

- Styling an application with skins
- Customizing an application appearance with skins
- Controlling the application appearance with a skin

## LESSON Styling an Application with Skins

Branding plays a vital role in presenting a uniform appearance across an application. Well-designed formatting and styles help guide users through navigation. Application developers change the look and feel of an application and enhance the user experience.

### **After this lesson, you should be able to:**

- Explain the differences between a skin, mixin, and format
- Define the mixin categories

# Skins

- A **skin** defines the responsive behavior and formatting, such as colors, fonts, images, and layouts used in a Pega application.
- A skin also defines the responsive breakpoints applied to dynamic layouts.
- Responsive breakpoints enable your application to work on various devices, such as tablets and mobile phones.
- A skin applies formatting through the use of mixins and formats.

# Mixins

- A  **mixin** defines a set of style attributes that can be reused in user interface elements.
- Mixins allow you to define efficient and clean style repetitions, and provide an easy way for you to update styling.
- Mixins can either define a set of styles or inherit styles from another mixin.

Repeating layouts reference a page list or page group, either by referencing a clipboard property or a data page.

# Mixin Categories

Mixins can either define a set of styles or inherit styles from another mixin. You can define four categories of mixins, listed in the following table.

Category	Description
Typography	Allows you to configure anything related to text, like font size, or color
Background	Allows you to configure background colors of elements
Border	Allows you to configure borders and gradient effects
Combination	Allows users to create complex mixins that incorporate multiple mixin types

Repeating layouts reference a page list or page group, either by referencing a clipboard property or a data page.

# Formats

- A **format** defines the styling of a specific UI component.
- A component is an element that you can style within the skin. For example, a layout (dynamic layout) or a control (button).
- You configure a format by setting the properties in the format or by inheriting styles from a mixin.
- When you update the mixin, any format based on the mixin is automatically updated.

# Formats Categories List

The following table lists the four categories of components where you add formats.

Category	Description
General	Modal dialogs, Errors
Layouts	Dynamic Layouts, Trees and Grids
Controls	Buttons, Dropdowns, Labels
Reports	List View, Paging Bar

## LESSON

# Customizing an Application with Skins

Skins are applied at an application level but can also be applied to a portal. A best practice is to reuse the application skin in any portal. However, at times you may create a custom skin for a portal. Application developers apply styles to various data elements.

### **After this lesson, you should be able to:**

- Explain the difference between a skin and mixin inheritance
- Identify the advantage of using skin inheritance

# Skin Inheritance

- Every application defines a skin used for the UI. The formats are defined in the skin.
- Before creating a skin, determine if a skin already exists that has a set of base styles from which you can inherit.
- Skin inheritance allows the skin to inherit formats and mixins from the parent skin.
- An advantage of using skin inheritance is that you can create an enterprise-wide layering of styles.

# Mixin Inheritance

- Mixins are the basic skin element for customizing the look and feel of an application.
- Mixins can inherit from other mixins, allowing you to broadly define styles and customize those styles as necessary. A mixin for errors can inherit styling from a base notification mixin, then customize the background color.

Notify

Error

- Changes to the base mixin affect any mixin that inherits from the base mixin.



DEMO

## Controlling Application Appearance with a Skin



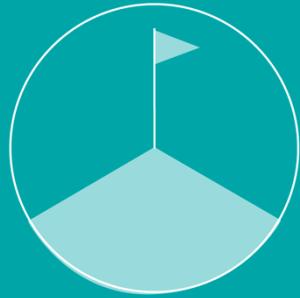
## GROUP DISCUSSION

© 2018 Pegasystems Inc.

12

What element should you create to maintain a consistent group of styles?





# CAPSTONE EXERCISE

## Designing a User Interface

© 2018 Pegasystems Inc.

## MODULE Sending Correspondence

This module contains the following lessons:

- Case communications
- Sending an email from a case
- Configuring correspondence rules

Organizations depend on timely communication to establish a shared understanding of transactions or assignments. You may also have a requirement to keep those directly and indirectly involved in the case up-to-date.

**After this lesson, you should be able to:**

- Explain how correspondence improves a process
- Identify users to communicate with
- Identify how and when to communicate with users

# Identifying Users to Communicate With

Pega defines the following **roles** for correspondence:

- Owner - The person who created the case.
- Customer - The person on whose behalf the case is transacted.
- Interested - A person who tracks the progress of a case but does not process the case.

Pega allows you to choose a role or a **party**.

# How and When to Communicate with Users

Pega provides four correspondence types:

- Email
- Text Message
- Fax
- Mail

Pega simplifies sending a correspondence by allowing you to simply add a step to your case.

In Pega, you add a Send Email step to your case and then configure the step to send email from a case.

**After this lesson, you should be able to:**

- Describe the process for creating correspondence
- Add email correspondence to a case type



DEMO

## Add and Configure a Send Email Step

## LESSON

# Configuring Correspondence Rules

Create correspondence rules to define, in HTML, templates for the content of outgoing correspondence. Each correspondence rule contains text for one type of correspondence such as email, letter, SMS phone text, or fax.

**After this lesson, you should be able to:**

- Identify correspondence types
- Add correspondence to a case type
- Send correspondence while processing a case
- Incorporate case content into correspondence

# Identify the Correspondence Type

A **correspondence type** indicates whether a piece of correspondence is a:

- Printed letter
- Fax
- Email
- SMS phone text

Each type is associated with a different **Data-** subclass, such as **Data-Corr-Email**, that holds the content of correspondence items.



DEMO

## Send Email Using the Send Email Step

© 2018 Pegasystems Inc.

9

To add a Send Email step to the case life cycle, select **Automations > Send Email**, and then click **Select**.



## DEMO

### Send Assignment Notification Emails

© 2018 Pegasystems Inc.

10

To enable assignment notifications for a case type, click the **Settings** tab and select the **Notifications** tab, then enable the **Email user when assignment is routed to worklist** option, using either App Studio or Dev Studio.



## DEMO

### Send Other Types of Correspondence

© 2018 Pegasystems Inc.

11

When you add a Utility shape to your process, open the properties panel and select CorrNew in the **Rule** field. Select your correspondence rule in the **CorrName** field. When a case reaches the utility, the system sends the correspondence.



DEMO

## Add Content to Correspondence



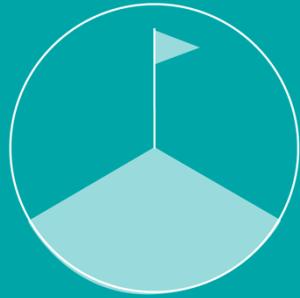
## GROUP DISCUSSION

© 2018 Pegasystems Inc.

13

- To achieve effective communication, what three questions should be addressed?
- What are the Pega defined roles?
- What four correspondence types are available?
- How do you use the Send Email step?
- How do you incorporate case content into correspondence?
- What is the relationship between a correspondence type and a correspondence rule?





## CAPSTONE EXERCISE

Creating and Sending Correspondence

© 2018 Pegasystems Inc.

## MODULE Creating Business Reports

This module contains the following lessons:

- The role of reports
- Business and process reports
- The Report Browser
- Creating a report
- Organizing a report

When designing reports, knowing what information the user needs and how it will be used is important. Pega Platform provides reporting features for developers, business analysts, and managers so that you can create simple or complex reports to satisfy your business needs.

**After this lesson, you should be able to:**

- Describe the role of reports in Pega applications
- Illustrate how report definitions retrieve records from a database
- Explain how report columns format data value
- Describe how functions make reports more useful
- Explain how report filters compare data value

# Role of Reports

Pega reporting capabilities allow you to create reports that provide real-time information.

Real-time information is important for two reasons:

- To assess the performance of an application.
- To view and select items from a list or table while working in an assignment.

# Report Definitions

- **Report definition** is used to build a report.
- The definition retrieves records from a database and returns the results in a table of columns and rows.

The diagram shows a table with 7 rows and 4 columns. The columns are labeled 'Employee' (containing O-101, O-104, O-746, O-983, O-171, O-623, O-421), 'Hire Date' (containing James Martin, Anne Walker, Julia Phagan, William Kirk, Leonard Kelley, Kate Picardo, Robert Wang), and 'Atlanta' (containing 2/21/16, 2/4/16, 1/12/16, 9/8/16, 9/5/16, 2/25/16, 2/1/16). A red box highlights the first two rows (O-101 and O-104). A white callout bubble points to the first row with the text 'Rows represent records from the database'. A blue box highlights the last three columns (Employee, Hire Date, Atlanta). A blue callout bubble points to the third column with the text 'Columns contain data values in each record'.

Employee	Hire Date	Atlanta
O-101	James Martin	2/21/16
O-104	Anne Walker	2/4/16
O-746	Julia Phagan	1/12/16
O-983	William Kirk	9/8/16
O-171	Leonard Kelley	9/5/16
O-623	Kate Picardo	2/25/16
O-421	Robert Wang	2/1/16

# Report Columns

- **Report columns** define the report's contents.
  - Each column corresponds to a single data element.
  - The column references a value property such as a case ID or work status.
  - You can format the data value in various ways, such as text, currency, or date.

# Functions

- **Functions** in columns make reports more useful.
- They allow users to calculate results derived from data in the database.
- Pega provides many standard functions to use in the report definition.

# Report Filters

- By default, report queries return all the records that contain data from all the columns.
- A **report filter** compares a data value in the record against a defined condition.
  - Filters limit results to relevant records.
  - Records that fail the filter comparison are omitted.

Case ID	Employee	Hire Date	Location
O-101	James Martin	2/21/16	Atlanta
O-104	Anne Walker	2/4/16	Boston
O-746	Julia Phagan	1/12/16	Atlanta
O-983	William Kirk	9/8/16	Atlanta
O-171	Leonard Kelley	9/5/16	Boston
O-623	Kate Picardo	2/25/16	Atlanta
O-421	Robert Wang	2/1/16	Boston



Case ID	Employee	Hire Date	Location
O-983	William Kirk	9/8/16	Atlanta
O-101	James Martin	2/21/16	Atlanta
O-746	Julia Phagan	1/12/16	Atlanta
O-623	Kate Picardo	2/25/16	Atlanta



# GROUP DISCUSSION

© 2018 Pegasystems Inc.

8

To reduce costs, business applications often target performance gains in time and efficiency. A business report that provides relevant information can show what is happening, what has happened over a period of time, and how what is happening matches or differs from your plan.

**After this lesson, you should be able to:**

- Define the two types of metrics associated with report data
- Differentiate between business data and process data in reports

# Business Reports and Process Reports

- There are two types of metrics associated with report data:
  - **Business metrics** that represent the data you define for an application.
  - **Process metrics** that are defined and tracked by Pega.

# Business Reports

What is being measured?	What is the data?	What is the business decision?
Average profit margin for all automobile sales last year	The average margin is below the target percentage.	The sales department decides to train its sales staff on promoting cars and options that have the highest margins.
Number of auto loans made in a month as compared to personal loans for the same period	The number of personal loans is significantly lower than the number of auto loans.	The goal is to have the numbers approximately equal. The marketing department increases marketing resources for personal loans.
Number of office desks shipped each week for the past month and how many are now in inventory	The number of orders shows an upward trend.	As a result, inventory levels are unacceptably low. The purchasing department decides to restock more desks on a weekly basis.

This table gives examples of business report information and how the information can be used in business decisions.

# Process Reports

What is being measured?	What is the data?	What is the business decision?
Open loan application cases exceeding the standard three-day service level deadline	Most of the open cases are for loan amounts greater than USD 300,000.	Loan requests that exceed this amount must go through an additional review step, which accounts for the delay. The department manager decides to increase the service level deadline for loans exceeding 300,000 USD from 3 to 4 days.
Average duration of assignments by type and action	This report identifies which user actions take the longest to complete.	Spend time on improving the efficiency of those assignments taking the most amount of time to complete.

© 2018 Pegasystems Inc.

12

The following table gives examples of process report information and how the information can be used in process design decisions.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

13

A sales manager runs weekly reports on how long it takes to prepare a car for customer delivery. Which type of report metrics apply to this report?

From the Report Browser, you can complete many tasks, such as browsing and searching for existing reports, running and scheduling reports, creating and modifying reports, and sharing reports with colleagues.

**After this lesson, you should be able to:**

- Describe the use of the Report Browser
- Explain how reports are grouped

# The Report Browser

The **Report Browser** is used to organize, run, and share reports.

Pega organizes reports by **category**. A category defines an organizational framework for reports displayed in the Report Browser.

- Public category reports are available to all users.
- Private category reports are available to specific users.

When you create a report, you design the report in the Case Manager portal by creating the report definition rule and adding columns. You can also use the Case Manager portal to add filters and make the report available in the Report Browser.

**After this lesson, you should be able to:**

- Create a report definition rule
- Add columns
- Add a filter
- Browse for your report in the Report Browser

# Creating a Report

- Process for creating a report includes:
  - Create a report definition rule
  - Add columns to define the data to display
  - Add a filter to limit the records you want to display
  - Make the report available in the Report Browser



DEMO

## How to Create a Report

**After this lesson, you should be able to:**

- Summarize column results for a report
- Display summary report data in charts or graphs
- Sort data in a column for a report
- Group report results to group data in a specified column

# Organizing Report Results

- Summarizing results
- Visualizing summary results
- Sorting values in the columns
- Grouping results

# Summarizing Results

- Summary reports help users to analyze a large amount of data.
- Summary reports allow users to quickly identify key statistics.
- Users can “drill down” to another report with additional details.



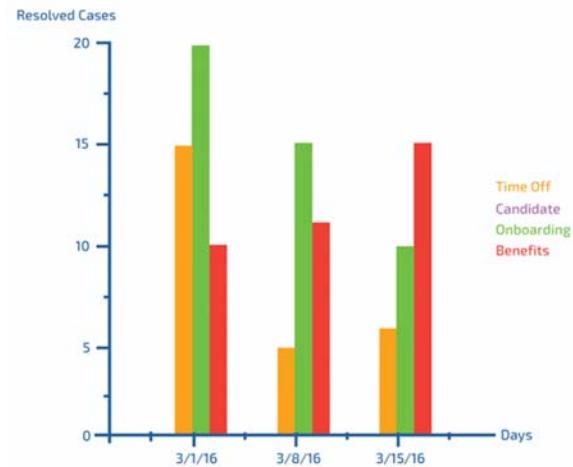
Manager	Case ID	Office
Harry	O-101	Atlanta
Fred	O-83	Boston
Fred	O-99	Boston
Harry	O-64	Atlanta
Harry	O-171	Atlanta
Harry	O-623	Boston

Manager	Case ID	Office
Harry	4	Atlanta
Fred	2	Boston

# Visualizing Summary Results

- Add a chart to present information in visual form.
  - Help users to quickly review and analyze data.
  - Add charts to summary reports.



# Sorting Values in the Columns

- Sort report results to organize the data in the report.
  - Sort column values in ascending or descending order
  - Specify a sort order to sort multiple columns

Employee	Start Date	Location
Kate Picardo	3/3/16	Atlanta
James Martin	2/9/16	Atlanta
Anne Walker	4/12/16	Boston
Robert Wang	2/9/16	Boston
Julia Phagan	3/8/16	Atlanta
William Kirk	2/8/16	Atlanta
Kelly Smith	4/5/16	Boston



Employee	Start Date	Location
Anne Walker	4/12/16	Boston
Kelly Smith	4/5/16	Boston
Julia Phagan	3/8/16	Atlanta
Kate Picardo	3/3/16	Atlanta
James Martin	2/9/16	Atlanta
Robert Wang	2/9/16	Boston
William Kirk	2/8/16	Atlanta

# Grouping Results

- Group results to organize the data in large list reports.

Start Date	Employee
Location: Boston	
4/12/16	Anne Walker
4/5/16	Kelly Smith
2/9/16	Robert Wang
Location: Atlanta	
3/3/16	Kate Picardo
2/9/16	James Martin
3/8/16	Julia Phagan
2/8/16	William Kirk



DEMO

## How to Organize Report Results



## MODULE Optimizing Report Data

This module contains the following lessons:

- Data storage in Pega applications
- Property optimization
- Optimizing properties for reporting

Pega applications allow application developers to improve report performance through a process called optimization. To store case data, Pega writes to a BLOB field, which provides greater flexibility and performance for case data.

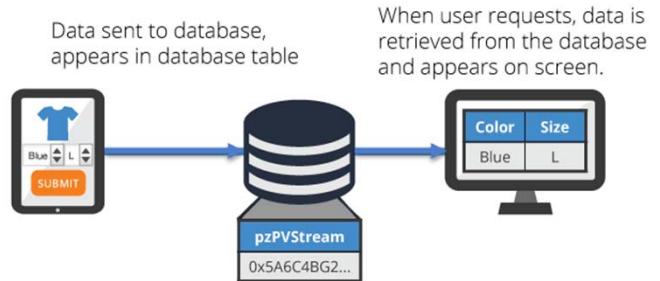
**After this lesson, you should be able to:**

- Describe how Pega stores case data
- Explain the advantages and disadvantages of using BLOB fields

# Data Storage

Pega Platform applications store each case as a unique record in a relational database in a **binary large object** (BLOB) field.

- Within the database table:
  - Each record is a row
  - Each column displays the contents of a field from the case record, including the BLOB field
- When an end user opens a case, Pega locates the record and reads the contents of the BLOB column of the table to extract the case data.



# BLOB Fields

- Advantages

- Unlimited storage size
- Flexibility
- High performance

- Disadvantages

- Penalizes performance
- Decompressing the BLOB fields significantly increases the time needed to run a report.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

5

Why is case data stored in a BLOB column?

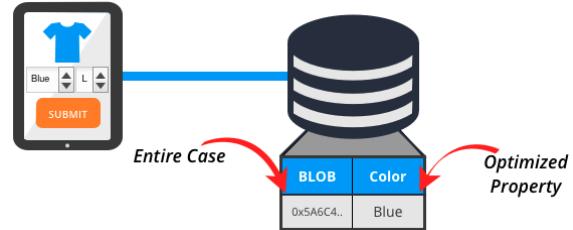
Extracting data from a BLOB impacts performance compared to reading property values from a database table. This impact is most pronounced when extracting data for report filters and sorting or grouping the content of a column. Pega's data storage model improves report performance with optimization.

**After this lesson, you should be able to:**

- Explain how property optimization affects properties
- Explain the impact of property optimization on report performance
- Describe where Pega writes data
- Describe how Pega reads data

# Property Optimization

- To improve report performance, Pega offers a hybrid data storage model
  - Data is stored both in dedicated indexed columns and in a BLOB field
  - Optimize the property for reporting to store property values in an indexed column



# Property Optimization (continued)

- When you optimize a property, Pega creates a column for the property in a database table.
- Optimizing a property for reporting is also called “exposing” the property.
- When a case or report uses an optimized property, Pega:
  - Writes data to both the property field and the BLOB field
  - Reads from the property column rather than the BLOB

# Property Optimization (continued)

- Optimization reduces the time and memory needed to run the report because decompressing does not occur.
- By default, Pega optimizes many properties that store process data. Some of these include:
  - The creation date of a case
  - The status of a case
  - The case ID
- Pega properties that store process data begin with the letters *px*, *py*, or *pz*.

# Property Optimization (continued)

- Properties created to store business data are not optimized by default.
- Reports that use an unoptimized property display a warning that states the potential impact on performance.
- These warnings due to an unoptimized property are resolved by running the Property Optimization tool.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

11

Why are properties exposed, or optimized, for reporting?

## LESSON

# Optimizing Properties for Reporting

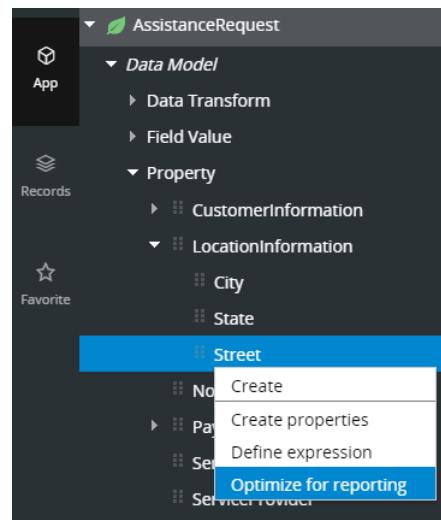
You can improve reporting and searching performance by optimizing, or exposing, Single Value properties as distinct columns. Pega provides the Property Optimization tool to optimize properties for reporting.

### **After this lesson, you should be able to:**

- Use the Property Optimization tool on a property to expose the property as a database column

# Optimizing Properties for Reporting

- Pega provides the Property Optimization tool to optimize properties for reporting
- It exposes the property as a database column and populates the new column by extracting values from the BLOB column.





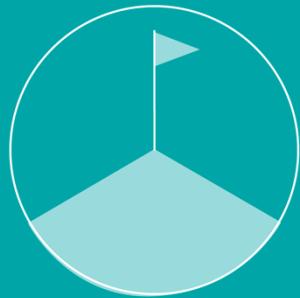
DEMO

## How to Optimize Properties for Reporting

© 2018 Pegasystems Inc.

14





# CAPSTONE EXERCISE

## Designing a Business Report

© 2018 Pegasystems Inc.

## MODULE Unit Testing Rules

This module contains the following lessons:

- Unit testing

Unit testing supports the continuous delivery of applications by enabling quality testing of individual rules. In this lesson, you learn how to unit test individual rules and record tests to run as part of automated tests.

**After this lesson, you should be able to:**

- Describe the role of unit testing in application development
- Test a rule with the Run Rule window
- Record a unit test to run at a later time

# Unit Testing

- **Unit testing** of individual rules is a basic form of application testing.
- Unit testing verifies that each element of the application works as expected.



# Unit Testing Reduces Configuration Errors

Consider a decision tree that evaluates a property. The application reads the property from a data page sourced from a report definition.

Unit testing individual rules ensures each rule works as expected.

- If the decision tree returns an incorrect result, but the data page contains the correct data, you can isolate the error to the decision tree.



# Unit Testing and DevOps

- Unit testing is a key enabler of a DevOps culture for application development.
  - DevOps relies on the automation of release management and packaging tasks to support a continuous development and continuous integration model of application development.
- Unit testing on a recurring basis can identify issues quickly.
- Developers can fix any issues before releasing an application, improving application quality.





DEMO

## Unit Test a Rule

© 2018 Pegasystems Inc.

6

Tip: Click **Reset Page** to clear the result of a previous test.



DEMO

## Record a Unit Test for Automated Testing



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

8

The purpose of unit testing is to \_\_\_\_\_.

The purpose of a test page is to \_\_\_\_\_.

What are two ways to view the output of a unit test?

What is the role of a test case in application development?

What is the relationship between an assertion and a test case?

What does the Unit testing landing page show?



## MODULE Delegating Rules

This module contains the following lessons:

- Business rule delegation
- Delegating rules to business rules

## LESSON

# Business Rule Delegation

Delegating business rules to business users enables the business to control their own policies. This enables the business applications to keep pace with changing business conditions.

**After this lesson, you should be able to:**

- Explain the benefits of delegating business rules
- Explain the purpose of delegating business rules
- Identify the types of business rules best suited for delegation to business users
- Identify the necessary details for delegating business rules
- Delegate a business rule to one or more business users

# Business Rule Delegation - Introduction

- Pega Platform enables you to design and implement applications that can respond to change with agility and efficiency.
- By recording business policies in rules rather than in code, an application can provide a degree of modularity and transparency that can simplify maintenance.
- Delegating rule changes to business users helps promote an agile response to continuously changing business conditions.



## GROUP DISCUSSION

© 2018 Pegasystems Inc.

4

What is the purpose of rule delegation?

# Business Rule Delegation – Best Practices

Establishing guidelines and best practices for rule delegation is critical to a successful rule delegation strategy.

- Establish a unique access group for the business users to whom you will delegate rules.
- Delegate rules for process items that change repeatedly.
- Delegate rules that are easy for the business user to change.

# Business Rule Delegation – Rule Selection

- You determine which rules to delegate based on business needs and ease of business user access.
- Selecting a specific rule type determines how to simplify the way a business user changes them.
  - For example, service level rules provide a set of choices for selection such as sending an email and transferring an assignment to another user.
- The Pega Platform rule types you can delegate are paragraph, decision table, data types that have data records, correspondence templates, and service level agreement (SLAs).



## GROUP DISCUSSION

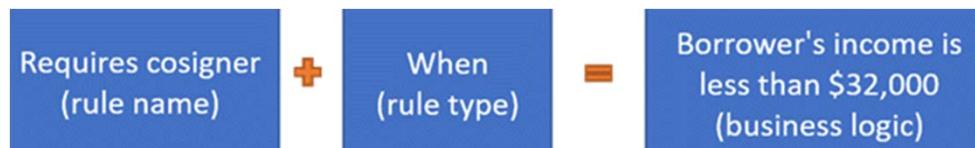
© 2018 Pegasystems Inc.

7

What are the criteria for determining rules to delegate?

# Business Rule Delegation – Rule Names

- Choose rule names that are meaningful in the business context.
- A good test of a rule name is to build a sentence using the rule name, the rule type, and the business logic.



# Delegating Rules to Business Users

- Create an access group for users who manage delegated rules
- Identify the rules to delegate
- Organize the rules to delegate in an unlocked, production ruleset
- Delegate the rule



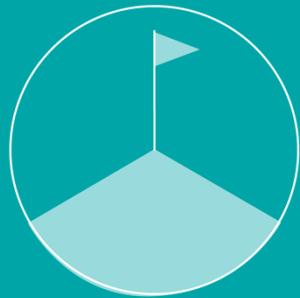
## GROUP DISCUSSION

© 2018 Pegasystems Inc.

10

What is a good test for a name for a delegated rule?





## CAPSTONE EXERCISE

### Testing and Maintaining an Application

© 2018 Pegasystems Inc.

# Course Summary

Now that you have completed this course, you should be able to do the following:

- Explain the benefits of using the Pega model-driven application design and development approach
- Model the life cycle of a case that mirrors the way business people think about how work is completed
- Identify the high-level responsibilities associated with Pega Platform for both Pega business architects and system architects
- Describe Pega's Direct Capture of Objectives approach to increasing the speed and accuracy of application delivery

# Course Summary (continued)

- Explain the purpose and benefits of best practices and guardrails
- Validate case data to ensure that user entries match required patterns
- Configure a Wait shape to enforce a case processing dependency
- Configure user views and data elements during case life cycle creation
- Use the Clipboard tool to review case data in memory
- Set property values automatically using data transforms and declare expressions
- Configure and populate a work party with case data

# Course Summary (continued)

- Create data classes and properties for use in a Pega application
- Automate decision-making to improve process efficiency
- Design responsive user forms for use on any platform or browser
- Design reports to deliver key insights to business users
- Incorporate and manage reference data to allow applications to adapt to changing business conditions
- Test your application design to analyze rule behavior and identify configuration errors