SKILL LESSON

Classes and class hierarchy

Understanding rule reuse through class inheritance

Overview

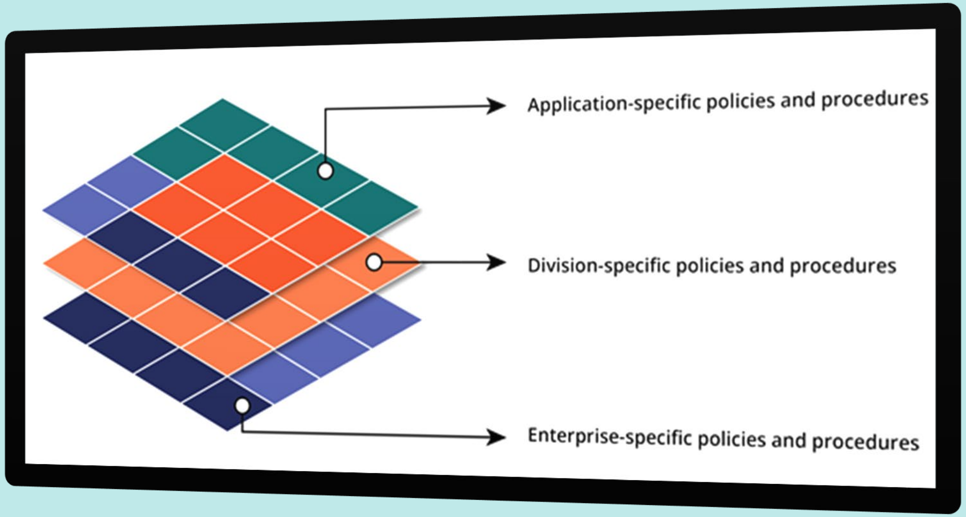
Pega Platform™ organizes application rules into classes, which can be reused.

Classes are layered into a multi-level hierarchy to organize and share application assets, data, and rules.

Inheritance allows applications to reuse existing rules or guidelines in other cases, decreasing developing time while maintaining quality.

This module covers:

- Classes, class types, and hierarchy
- The class structure layers
- Rule reuse through inheritance


Application-specific policies and procedures
Division-specific policies and procedures
Enterprise-specific policies and procedures

Pega vs object-oriented programming

The usage of a class in Pega is like object-oriented programming (OOP) where a class contains properties and methods.

In Pega, you add the following to a :

Properties of different types:

- Single Value (Text, Decimal, Boolean, Date)
- Data Relationship (a common context for a group of fields; single-an instance of another class; multiple-contains multiple objects)

Methods that have a specific usage:

- Section / Paragraph / …
- Report Definition
- Validate / Edit validate / … class

In OOP, you add the following to a class:

Properties of different types:

- Single Value (Text, Decimal, Boolean, Date)
- Object (an instance of another class)
- List/Map of Objects (contains multiple objects)

Methods that have a specific usage:

- Render UI
- Fetch data from a data base
- Data validation

Classes and class types

Classes

Pega enables developers to reuse rules across different types of cases and applications, which enhances the quality of the applications and reduces the time it takes to create them.

When creating an app, Pega organizes the rules into classes based on their ability to be reused. Each of these groups is known as a class. Each application consists of three class types:

Work class (Work-)

Contains the rules that describe how to process cases.

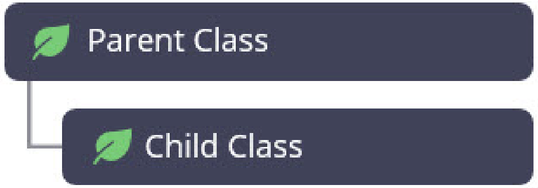
Integration class (Int-)

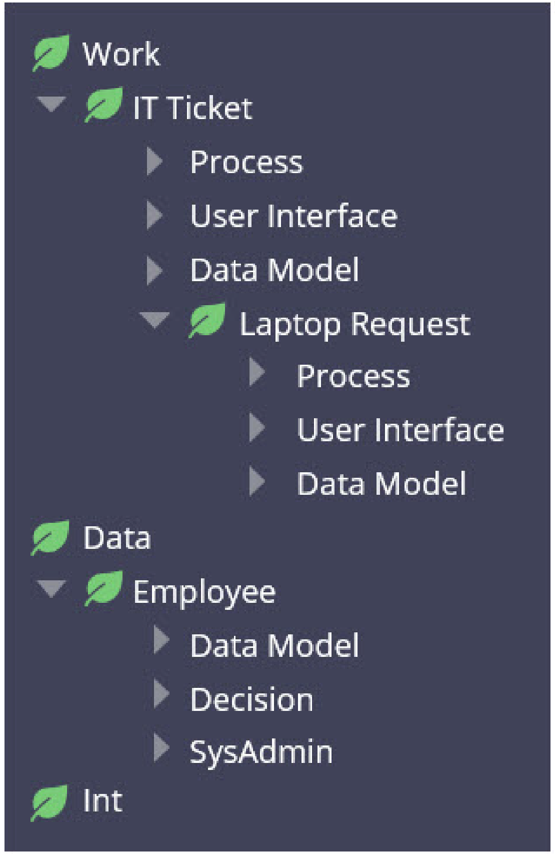
Contains the rules that describe how the application interacts with other systems.

Data class (Data-)

Contains the rules that describe the data types in the application.

Class types


Parent Class
Child Class

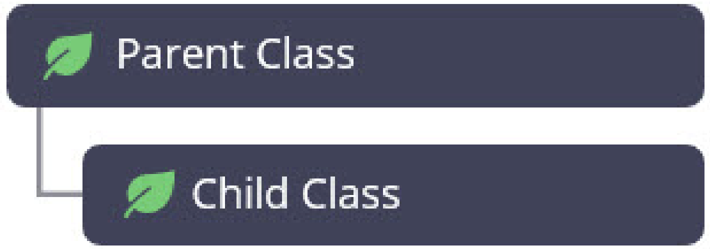

Work
IT Ticket
Process
User Interface
Data Model
Laptop Request
Process
User Interface
Data Model
Data
Employee
Data Model
Decision
SysAdmin
Int

Class types

Parent and child classes

A class that contains another class, is a parent class, while any classes contained within the parent class, is a child class.

Child classes can reuse or inherit any rules that are defined for its parent class.

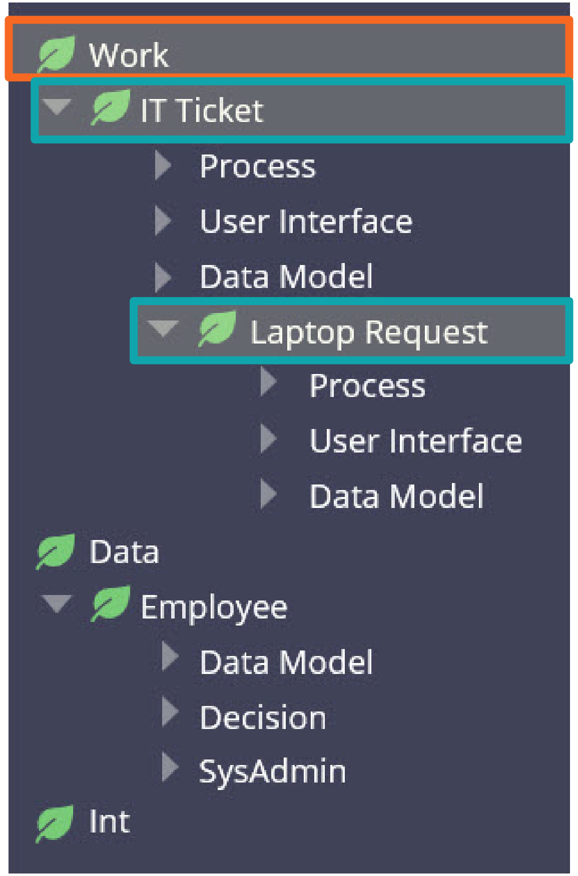


Class types

Work class

Contains a child class for each case type in your application. Each case type child class contains all the rules that are unique to a case type.

For example: the Work class contains the case types IT Ticket and Laptop Request.



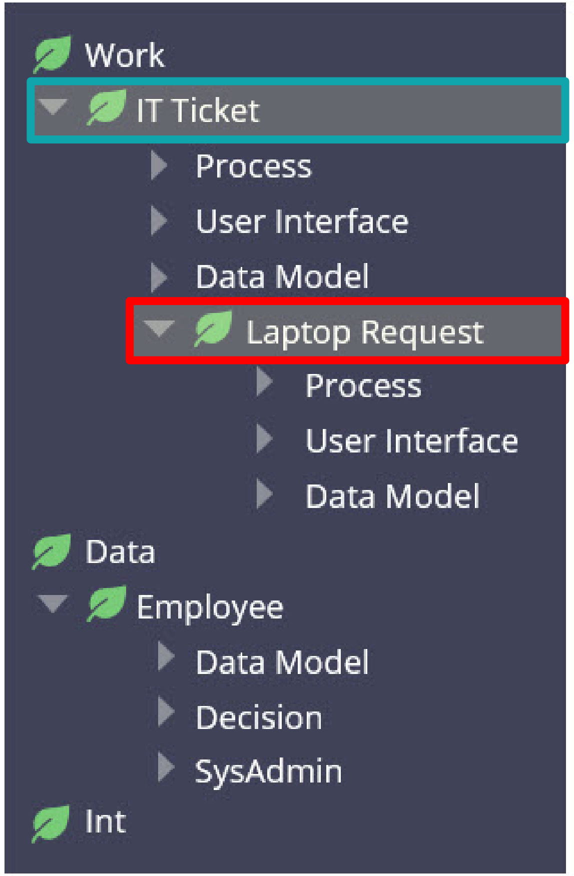
Class types

Parent case type

The IT Ticket case type class is the parent class of the Laptop Request case type class.

Child case type

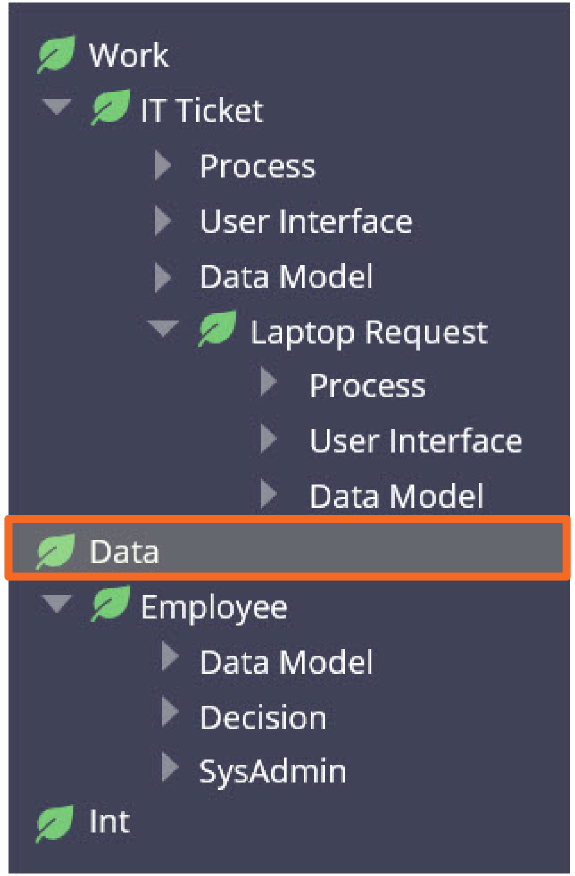
The Laptop Request case type class is a child class of the IT Ticket case type class. The Laptop Request is treated as a specialized version of a generic IT ticket.



Class types

Data class

The Data class contains the rules of each data type that is used in your application.



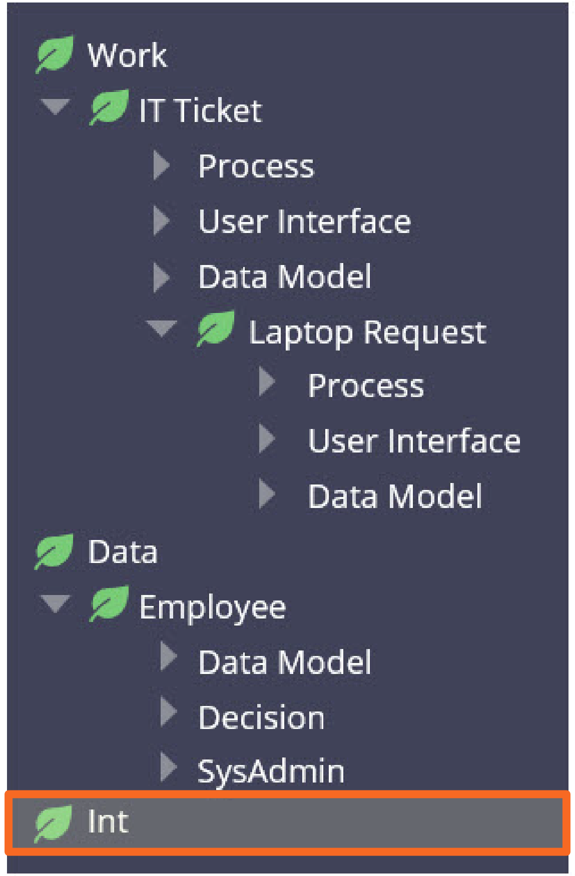
Class types

Data class

The Data class contains the rules of each data type that is used in your application.

Int class

The Int class contains the Integration rules such as SOAP and REST connectors for accessing data from external data sources.

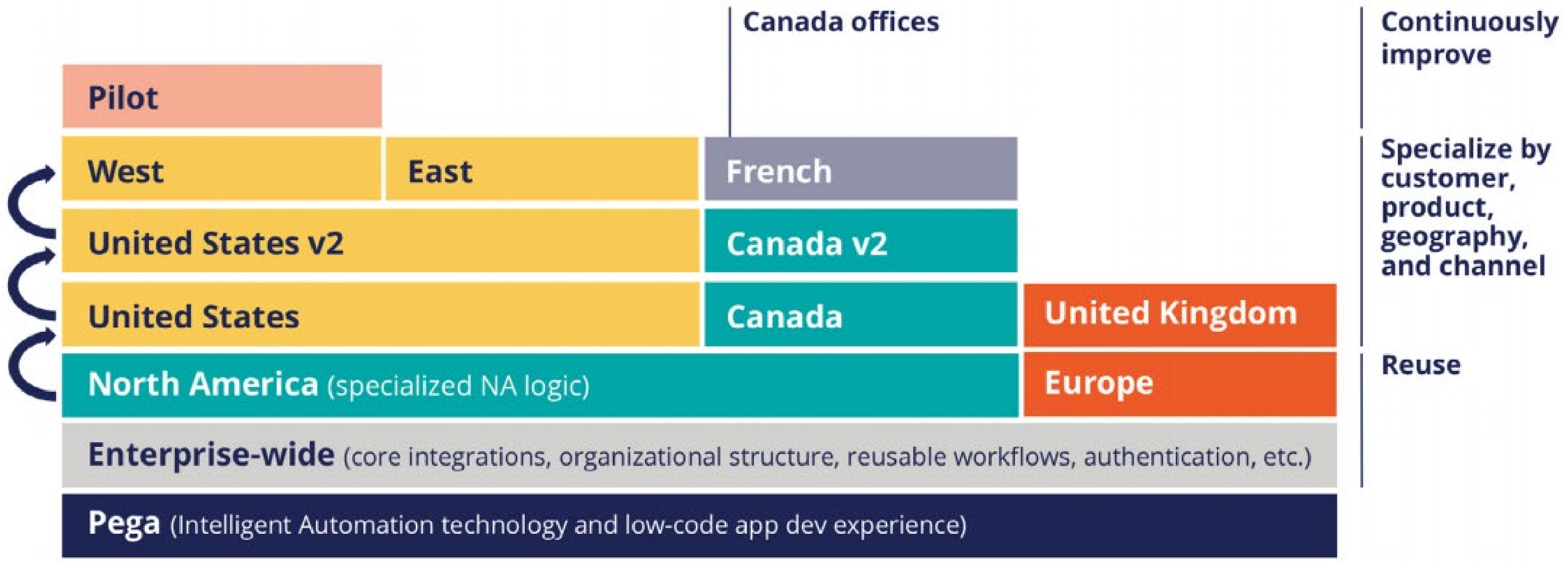

Work
IT Ticket
Process
User Interface
Data Model
Laptop Request
Process
User Interface
Data Model
Data
Employee
Data Model
Decision
SysAdmin
Int

Enterprise class structure and class hierarchy

Enterprise class structure

Pega provides an organizational model, called enterprise class structure (ECS), this allows you to:

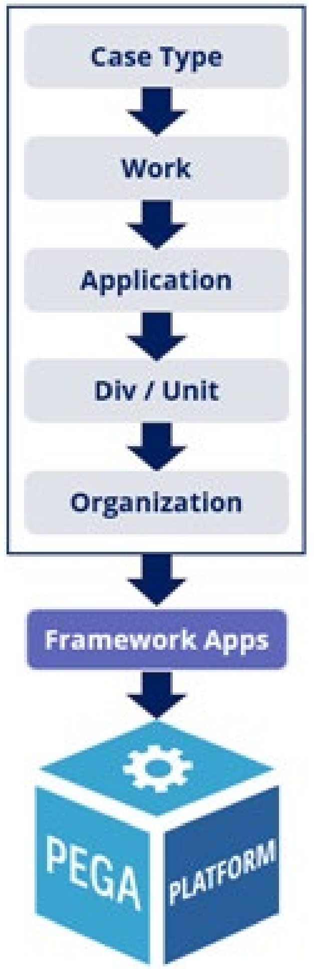
- Modify your application to suit the requirements of your business
- Implement the reuse of rules through inheritance



Canada offices
Continuously improve
Pilot
West
East
French
Specialize by customer, product, geography, and channel
United States v2
Canada v2
United States
Canada
United Kingdom
North America (specialized NA logic)
Europe
Reuse
Enterprise-wide (core integrations, organizational structure, reusable workflows, authentication, etc.)
Pega (Intelligent Automation technology and low-code app dev experience)

Class hierarchy

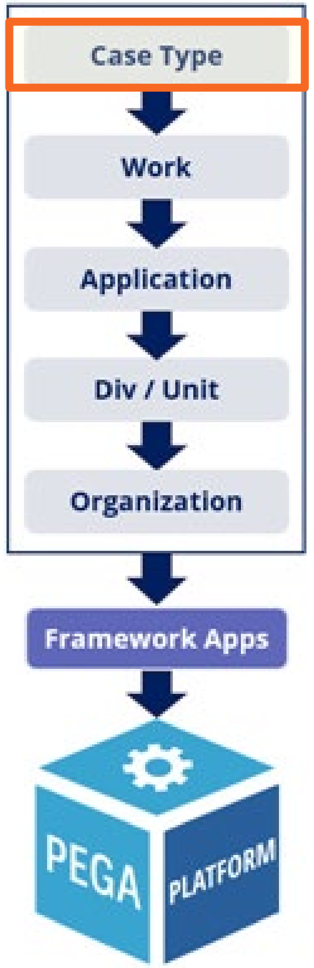
- Classes are layered into a multi-level hierarchy to organize and share application assets, data, and rules.
- Creating a class hierarchy allows you to determine which classes can inherit reuse already-established rules.
- Classes are searched upwards in the hierarchy when looking for already-established rules.
- Base classes are the immediate child classes of the hierarchy’s ultimate base class.
- The names of classes that belong to a common class group start with the group name, followed by a hyphen.



Class hierarchy

Case type classes

Describes a specific case type. The rules that are associated with case type classes are unique to the corresponding case type and cannot be reused elsewhere in the organization.



Class hierarchy

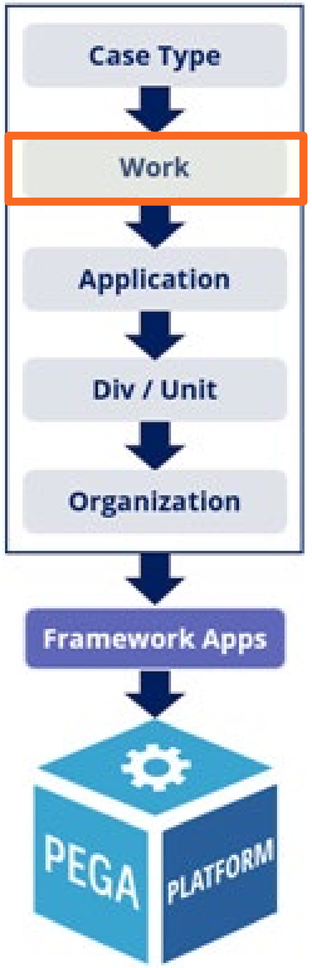
Case type classes

Describes a specific case type. The rules that are associated with case type classes are unique to the corresponding case type and cannot be reused elsewhere in the organization.

Work classes

Contains all case types in an application. In a Pega application, all case types are associated with a work class.

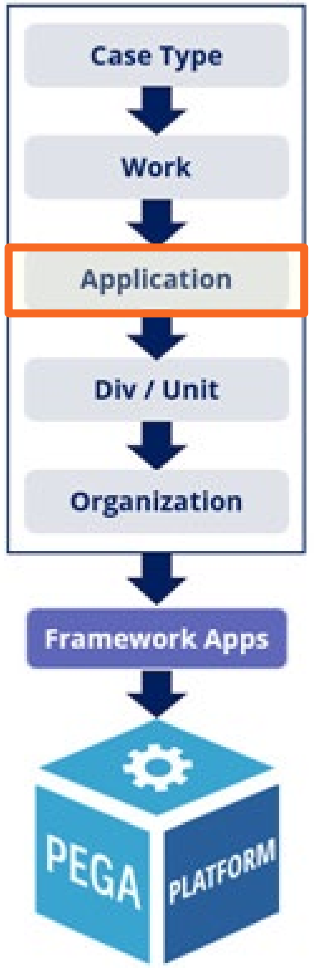
While integration and data classes can occur at any level of the class hierarchy, work classes are generally the most specific of the three classes.



Class hierarchy

Application classes

Contains the required work, integration, and data classes for a single application, and provides the ability to share common rules among these classes.



Class hierarchy

Application classes

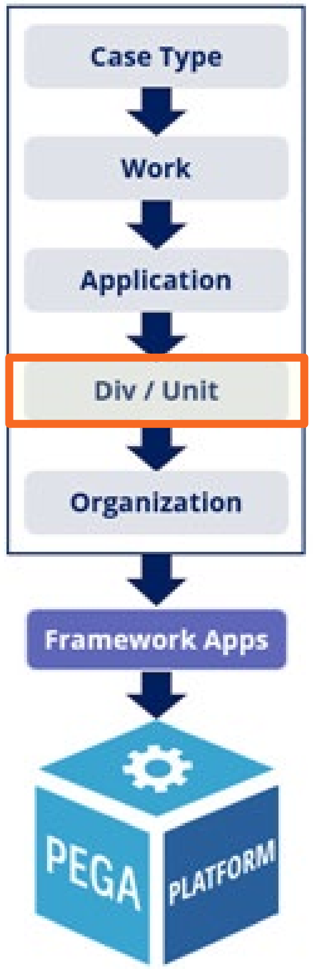
Contains the required work, integration, and data classes for a single application, and provides the ability to share common rules among these classes.

Division / Unit classes

The division layer contains the work, data, and integration classes used division-wide.

The optional unit layer contains the work, data, and integration classes for the unit.

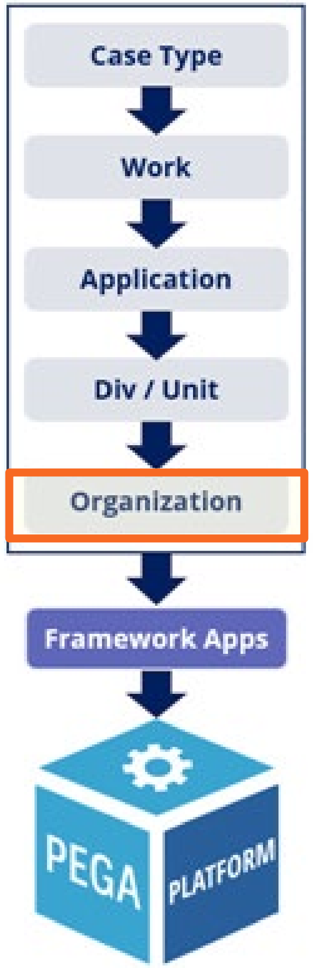
These classes allow the reuse of rules at the division and unit level and should only be created when necessary.



Class hierarchy

Organization classes

Contains all the classes for applications across an entire business or organization often containing data and integration classes.



Class hierarchy

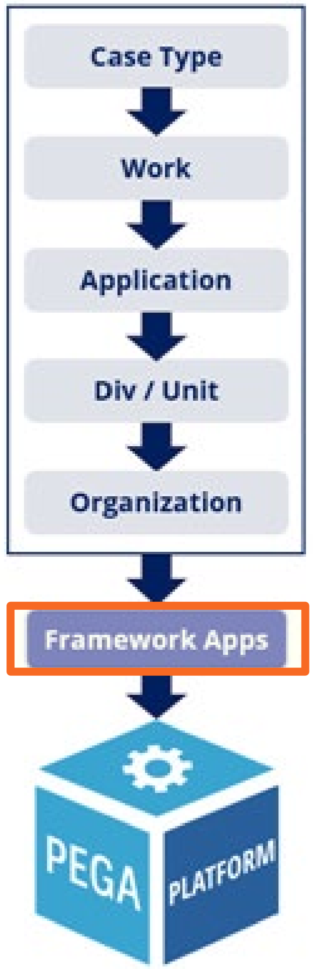
Organization classes

Contains all the classes for applications across an entire business or organization often containing data and integration classes.

Other application classes

Classes from other applications, such as industry-specific Pega applications, are contained in what is sometimes referred to as the framework layer.

For example: You use a generic application for ticket management administration as a baseline for customizing versions of different IT ticket types. Using other applications is optional and determined for each implementation by a more experienced developer.

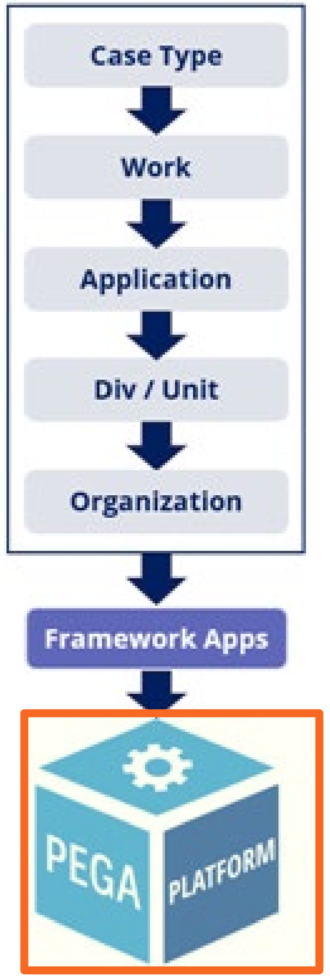


Class hierarchy

Base classes

Pega provides base classes with rules that provide basic functionality for case processing.

For example: You can access a base class with data elements that record who created a case and the time that is needed to complete an assignment.



Class inheritance and inheritance methods

Class inheritance

Inheritance allows an application to reuse existing rules for other cases or applications. Using rule inheritance decreases the time needed for development and assessment without decreasing the application's quality.

Pega supports the following types of class inheritance:

- Directed
- Pattern

Both types of class inheritance can adopt rules from ancestor classes. This doesn't apply to external classes, even ones that contain a hyphen.

Inheritance methods - Directed

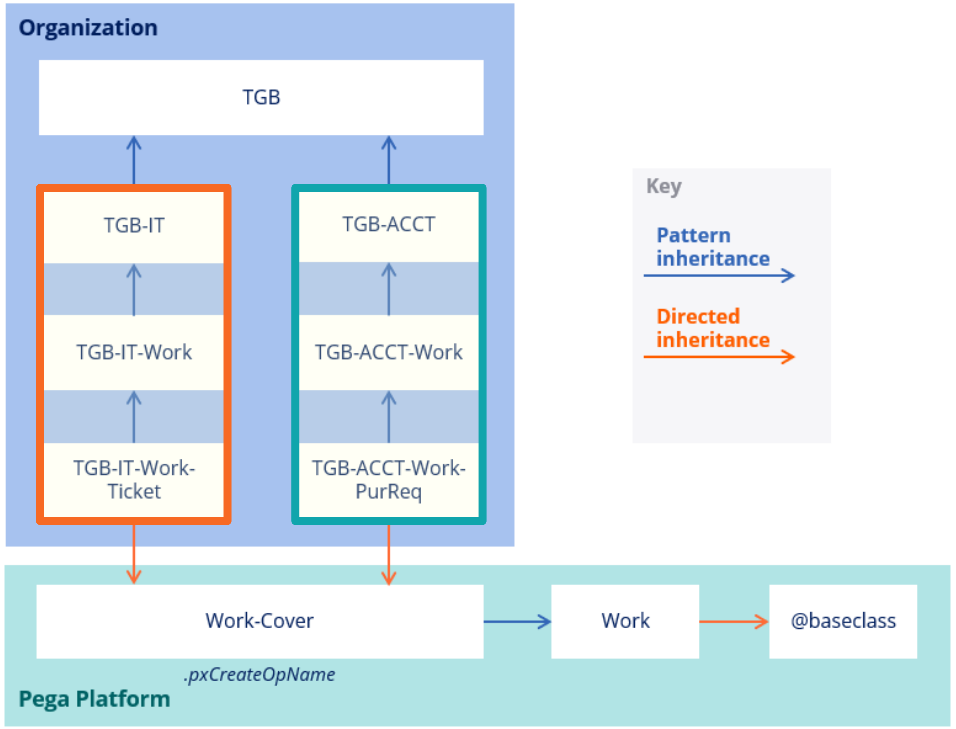
Unlike pattern inheritance, which occurs by default, directed inheritance involves specifying the parent class for inheritance explicitly.

For example: Consider an organization named TGB that has:

- An IT application with Ticket cases

and

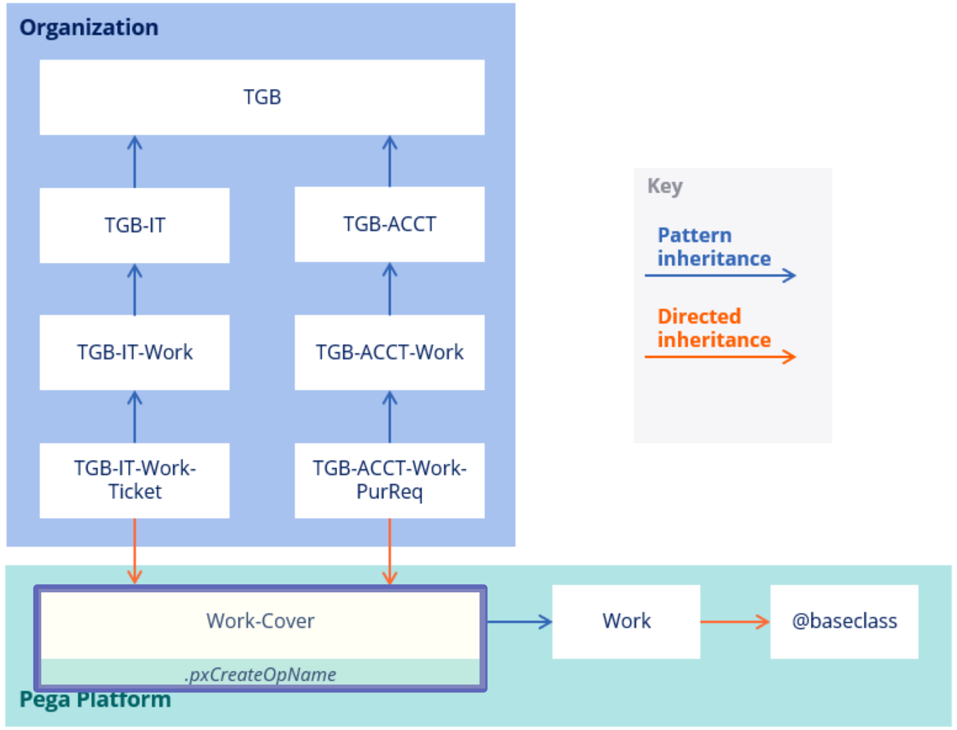
- An Accounting application with Purchase Request (PurReq) cases.



Inheritance methods - Directed

Unlike pattern inheritance, which occurs by default, directed inheritance involves specifying the parent class for inheritance explicitly.

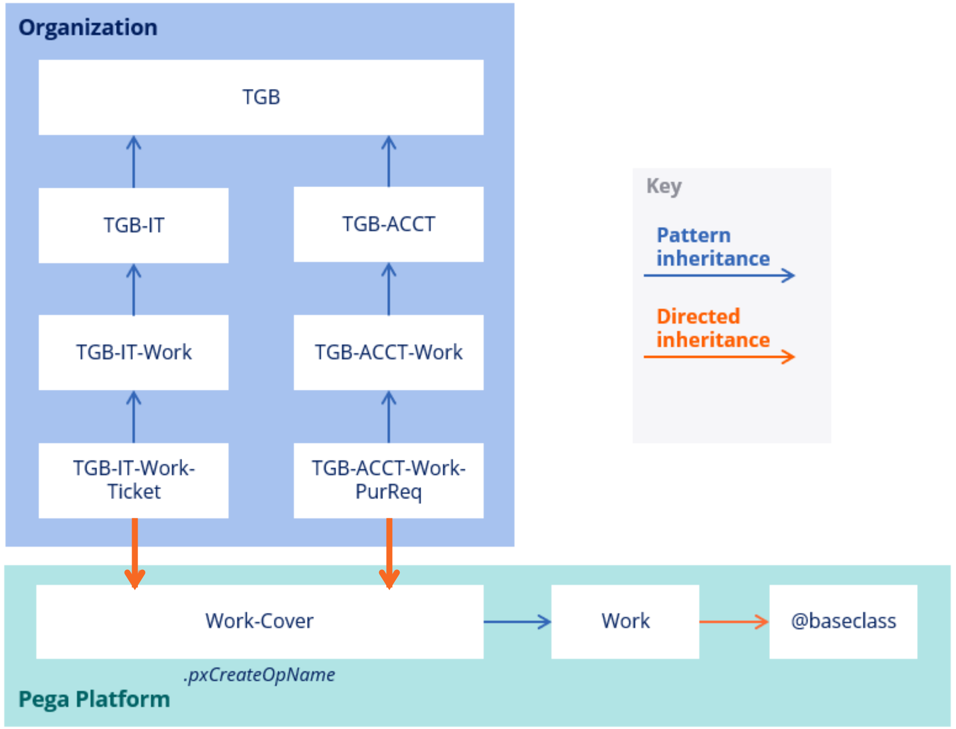
Both the Ticket and Purchase Request case types use a property, .pxCreateOpName, to record the name of the operator that creates the case.



Inheritance methods - Directed

Unlike pattern inheritance, which occurs by default, directed inheritance involves specifying the parent class for inheritance explicitly.

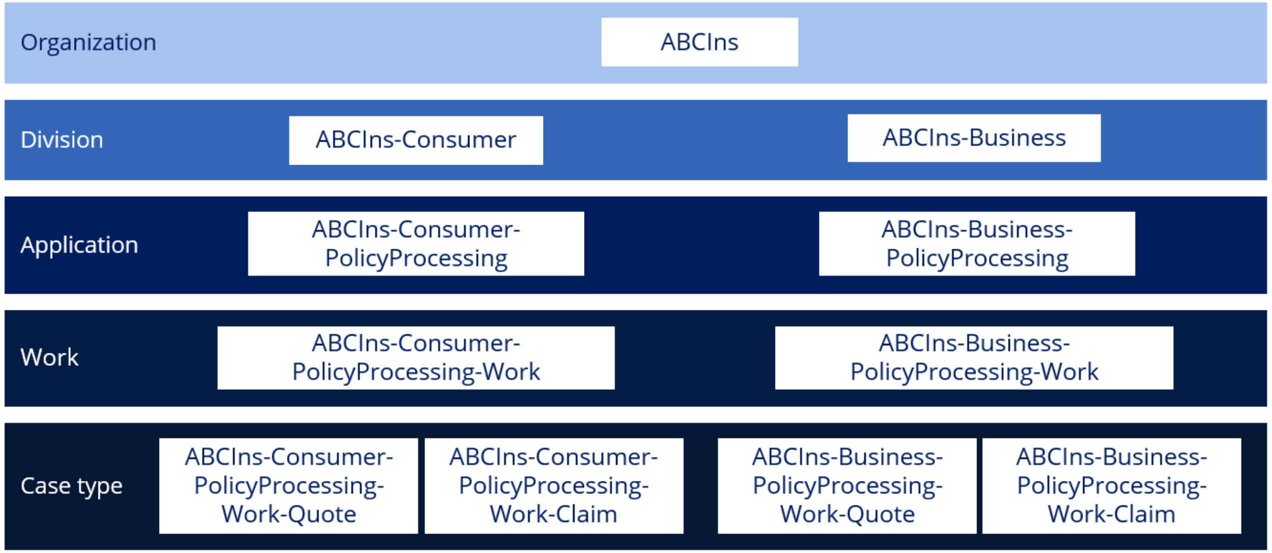
Through directed inheritance, both cases inherit the property which is defined in the Work-Cover class.



Inheritance methods - Pattern

Pattern inheritance is an automated system that uses the existing class name structure to decide which rules can be reused.

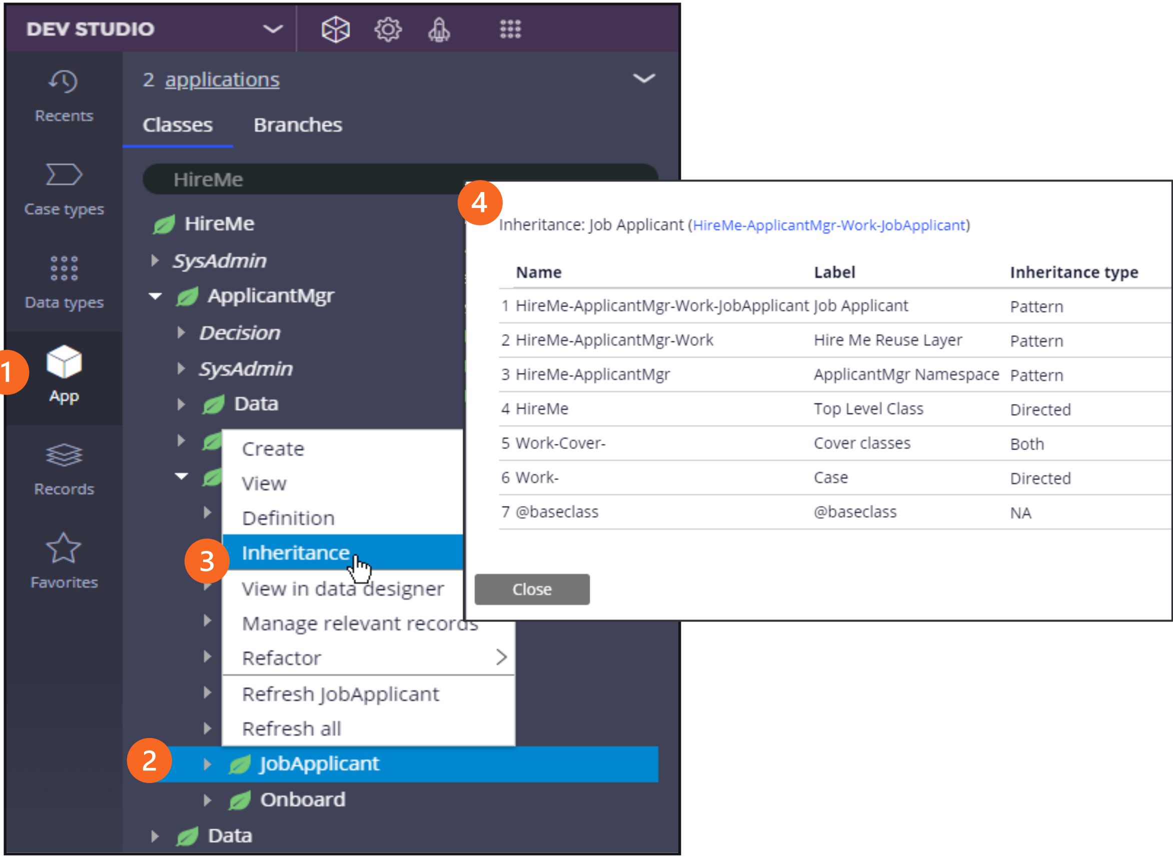
- Scans classes with same class prefix
- Promotes reuse based on business suitability


Organization
ABCIns
Division
ABCIns-Consumer
ABCIns-Business
Application
ABCIns-Consumer-PolicyProcessing
ABCIns-Business-PolicyProcessing
Work
ABCIns-Consumer-PolicyProcessing-Work
ABCIns-Business-PolicyProcessing-Work
Case type
ABCIns-Consumer-PolicyProcessing-Work-Quote
ABCIns-Consumer-PolicyProcessing-Work-Claim
ABCIns-Business-PolicyProcessing-Work-Quote
ABCIns-Business-PolicyProcessing-Work-Claim

Viewing inheritance structure

Dev Studio

1. In the navigation pane, click App.
2. Right-click a class.
3. Select Inheritance.
4. Review the inheritance path of a class.



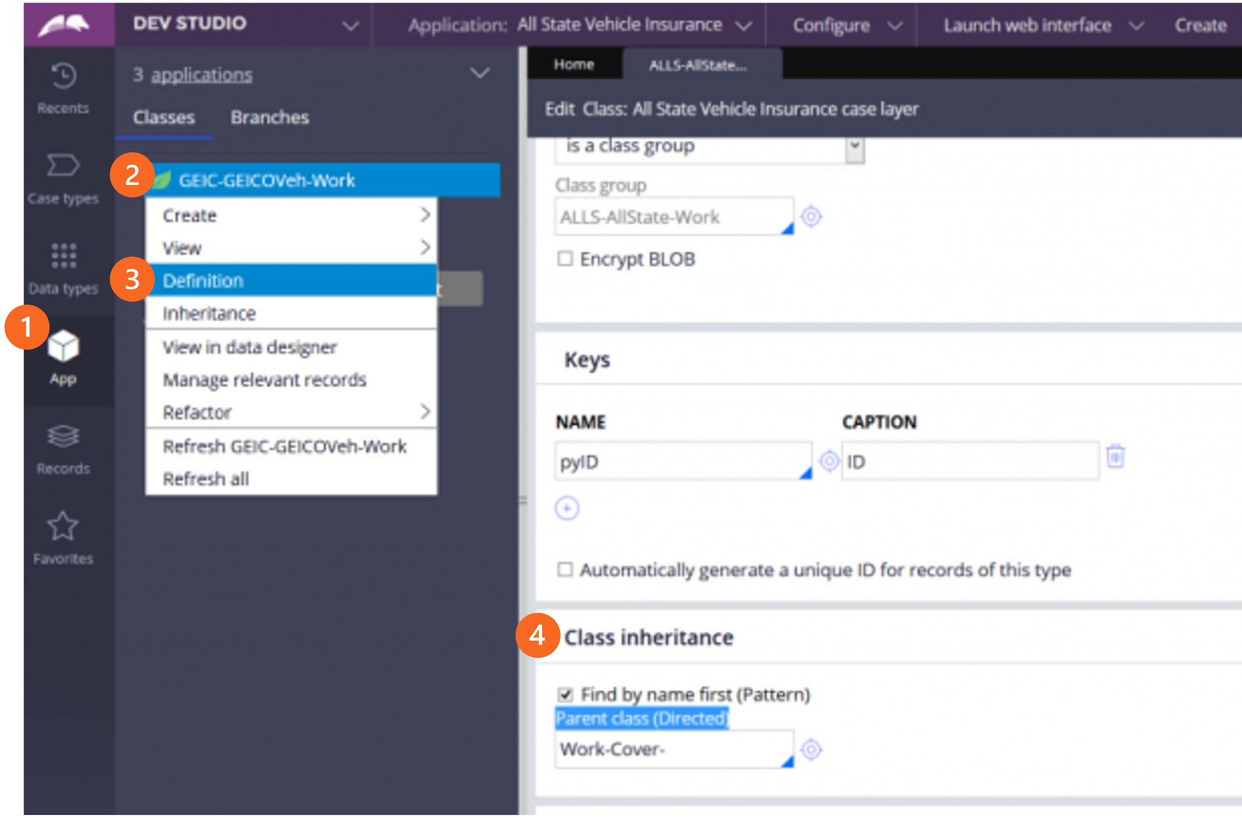
Inheritance: Job Applicant (HireMe-ApplicantMgr-Work-JobApplicant)

	Name	Label	Inheritance type
1	HireMe-ApplicantMgr-Work-JobApplicant	Job Applicant	Pattern
2	HireMe-ApplicantMgr-Work	Hire Me Reuse Layer	Pattern
3	HireMe-ApplicantMgr	ApplicantMgr Namespace	Pattern
4	HireMe	Top Level Class	Directed
5	Work-Cover-	Cover classes	Both
6	Work-	Case	Directed
7	@baseclass	@baseclass	NA

Update or review class inheritance

Dev Studio

1. In the navigation pane, click App.
2. Right-click a class.
3. Select Definition.
4. Update or review the class inheritance.



Edit Class: All State Vehicle Insurance case layer

is a class group

Class group: ALLS-AllState-Work

☐ Encrypt BLOB

Keys

NAME	CAPTION
pyID	ID

☐ Automatically generate a unique ID for records of this type

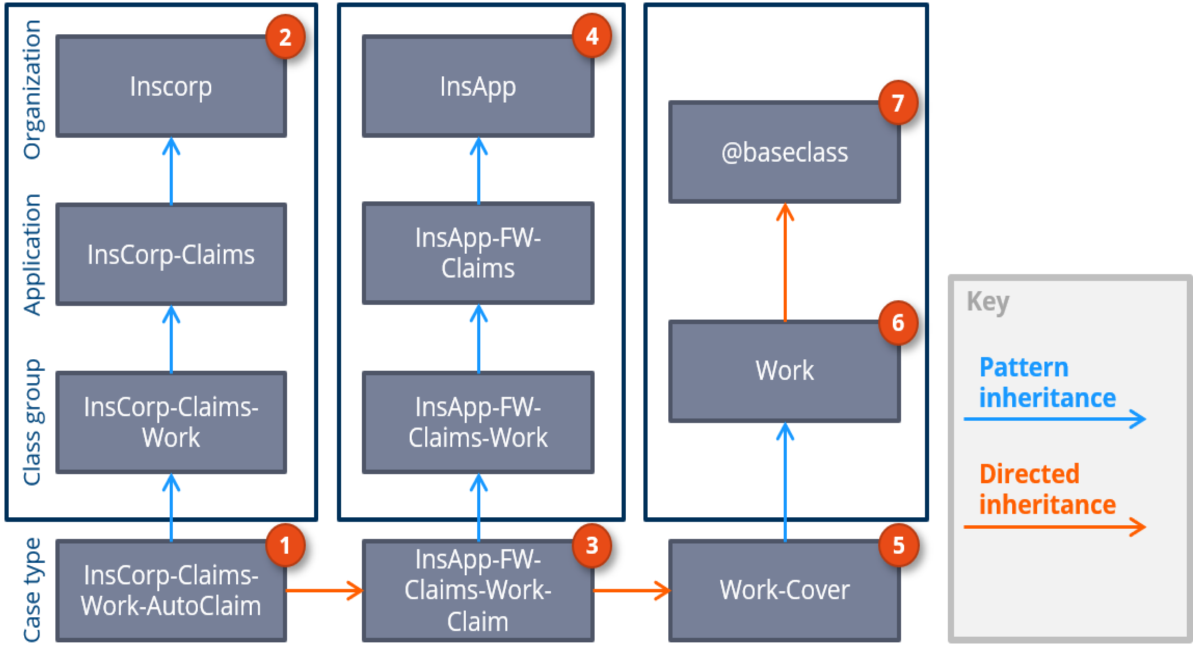
Class inheritance

☑ Find by name first (Pattern)

Parent class (Directed): Work-Cover-

Inheritance algorithm

- Pega searches through the parent classes indicated by pattern inheritance first, followed by directed inheritance.
- This process repeats until Pega reaches the ultimate base class.
- If the rule cannot be found after searching the base class, Pega returns an error.



Skill mastery

You understand:

- Classes and class types
- Enterprise class structure
- Class hierarchy
- Class inheritance and
- Inheritance methods

PEGA

PEGA®