A bean that has a bound property generates an event when the property is changed.
Bound Properties are the properties of a JavaBean that inform its listeners about changes in its values.
Bound Properties are implemented using the **PropertyChangeSupport** class and its methods.
Bound Properties are always registered with an external event listener.
The event is of type ***PropertyChangeEvent*** and is sent to objects that previously egistered an interest in receiving such notifications
bean with bound property - Event source
Bean implementing listener -- event target

In order to provide this notification service a JavaBean needs to have the following two methods:

```
public void addPropertyChangeListener(PropertyChangeListener p) {
        changes.addPropertyChangeListener(p);
    }
    public void  removePropertyChangeListener(PropertyChangeListener p) {
        changes.removePropertyChangeListener(p);
    }
```

PropertyChangeListener is an interface declared in the java.beans package. Observers which want to be notified of property changes have to implement this interface, which consists of only one method:

```
public interface PropertyChangeListener extends EventListener {
public void propertyChange(PropertyChangeEvent e );
    }
```

Constrained Properties:

Govind Kr. Rahul

Constrained Properties and Customizers and JavaBeans API

JAVA BEANS

VIGNAN'S LARA INSTITUTE OF TECHNOLOGY & SCIENCE

CONSTRAINED PROPERTIES

- A Bean that has a *constrained* property generates an event when an attempt is made to change its value.
- The event is of type **PropertyChangeEvent**. It is sent to objects that previously registered an interest in receiving such notifications.
- Those other objects have the ability to veto the proposed change.

0:17 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

PERSISTENCE

- *Persistence* is the ability to save a Bean to non-volatile storage and retrieve it at a later time. The information that is particularly important are the configuration settings.
- In the example of both the **Colors** and **TickTock** Beans, The **rectangular** property of the **Colors** Bean was changed to **true**, and the **interval** property of the **TickTock** Bean was changed to one second. These changes can be saved.

2:37 / 31:25

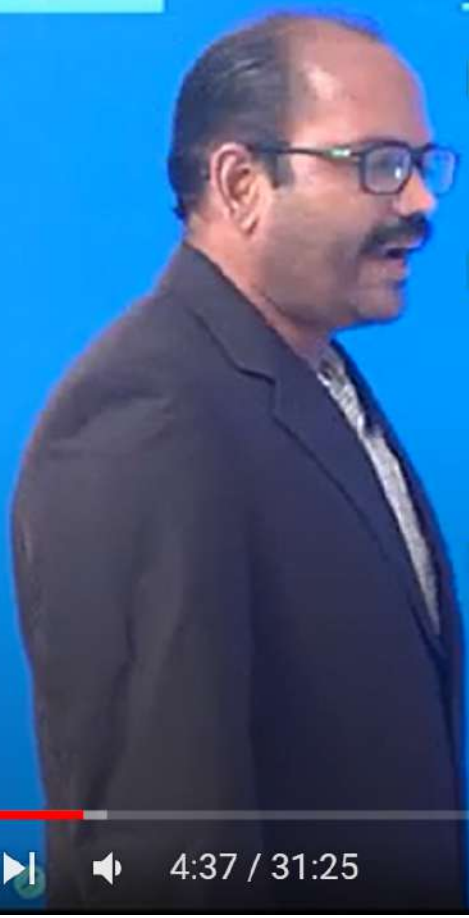
Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

PERSISTENCE

- To save the application, go to the menu bar of the BeanBox and select File | Save.
- A dialog box should appear, allowing to specify the name of a file to which the Beans and their configuration parameters should be saved.
- Supply a filename and click the OK button on that dialog box. Exit from the BDK.



4:37 / 31:25

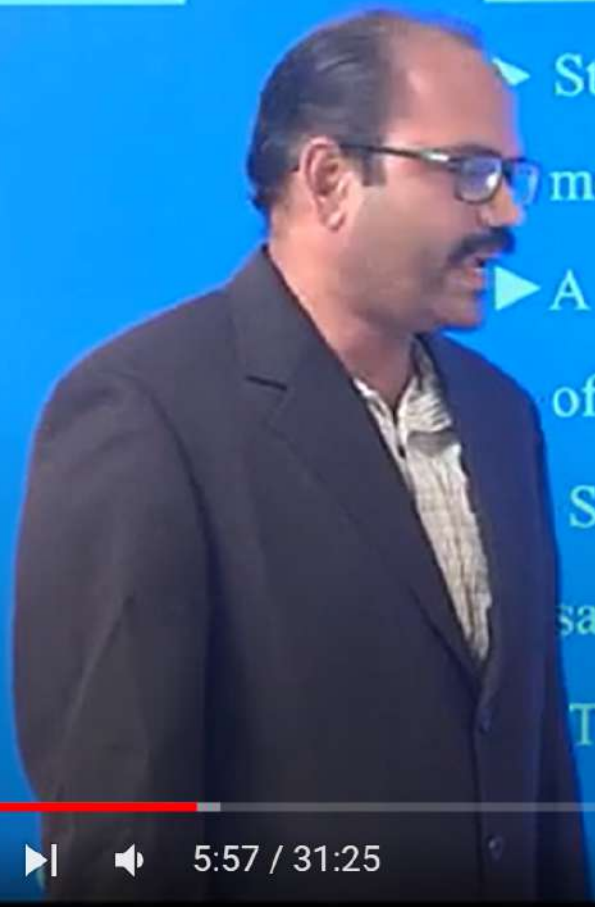
Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

PERSISTENCE

- Start the BDK again. To restore the application, go to the menu bar of the BeanBox and select File | Load.
- A dialog box should appear, allowing to specify the name of the file from which an application should be restored.
- Supply the name of the file in which the application was saved, and click the OK button.
- The application should now be functioning.



5:57 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

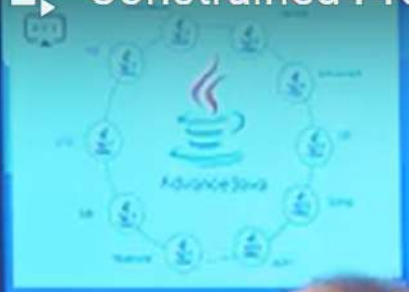
JAVA BEANS

PERSISTENCE

- Confirm that the **rectangular** property of the **Colors** Bean is **true** and that the **interval** property for the **TickTock** Bean is equal to one second.
- The object serialization capabilities provided by the Java class libraries are used to provide persistence for Beans.
- If a Bean inherits directly or indirectly from **java.awt.Component**, it is automatically serializable, because that class implements the **java.io.Serializable** interface.

7:07 / 31:25

Constrained Properties and Customizers and JavaBeans API

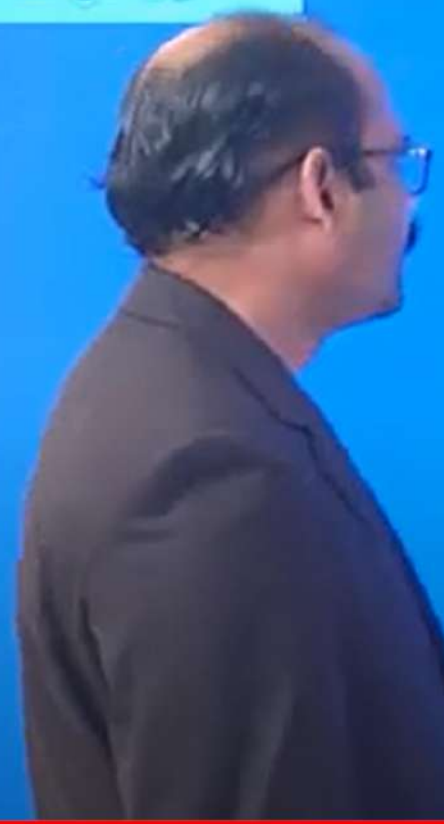


VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

PERSISTENCE

- If a Bean does not inherit an implementation of the Serializable interface, we must provide this.
- Otherwise, containers cannot save the configuration of the component.
- The transient keyword can be used to designate data members of a Bean that should not be serialized.
- The color variable of the Colors class is an example of such an item.



8:27 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

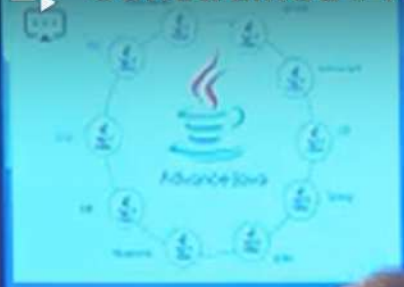
JAVA BEANS

CUSTOMIZERS

- The Properties window of the BDK allows a developer to modify the properties of a Bean.
- This may not be the best user interface for a complex component with many interrelated properties.
- Therefore, a Bean developer can provide a *customizer* that helps another developer configure this software.
- A customizer can provide a step-by-step guide through the process that must be followed to use the component in a specific context. Online documentation can also be provided.

10:27 / 31:25

Constrained Properties and Customizers and JavaBeans API

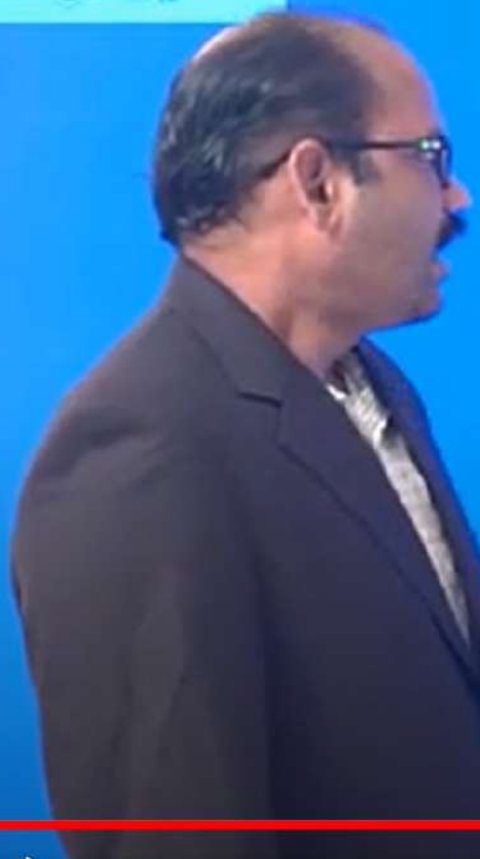


JAVA BEANS

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

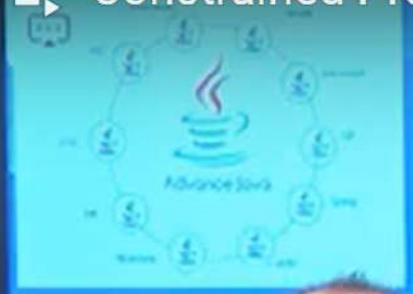
CUSTOMIZERS

- A Bean developer has great flexibility to develop a customizer that can differentiate his or her product in the marketplace.
- The JavaBeans model provides an alternative to property sheets.
- It specifies an interface called Customizer to enable bean authors to implement a bean-specific customizer.
- Such a customizer can be a simple panel with related properties grouped together or a sophisticated wizard that allows the user to navigate through a hierarchy of screens.



12:17 / 31:25

Constrained Properties and Customizers and JavaBeans API



VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

CUSTOMIZERS

- Since builder tools recognize any customizer that implements the Customizer interface, it can be launched from any builder tool that has the bean installed.
- **Customizer Interface :** The Customizer interface is simple and has only three methods.
- **setObject(Object bean):** The builder tool calls this method to pass the target bean instance to the customizer. It is called only once, which is before the builder tool launches the customizer.

14:07 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

CUSTOMIZERS

- ***addPropertyChangeListener(PropertyChangeListener listener):*** This method adds a propertyChange event listener. By registering as a listener, the target object(s) can receive notification and the modified value of the property. When the target object receives a propertyChange event, it retrieves the setter Method object and the property value from the PropertyChangeEvent object. Using this Method object and the property value, the target object then invokes the setter method in the bean to modify the property.

15:57 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

CUSTOMIZERS

- ***removePropertyChangeListener (PropertyChangeListener listener):*** This removes a property change event listener.
- A customizer class has to meet the following requirements to enable builder tools to launch it:
- ***It must implement the Customizer interface:*** This is to enable the builder tool to recognize the customizer and to obtain the bean instance, which can be passed on to the customizer. Using this bean instance, the customizer can fetch and modify a bean's properties.

17:47 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

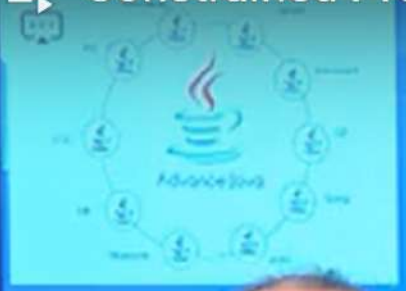
JAVA BEANS

CUSTOMIZERS

- ***It has to be a subclass (direct or derived) of the java.awt.Component class.*** This is to enable builder tools to embed the customizer in a dialog box.
- ***It must have a constructor with no arguments.*** This is to enable the builder tool to launch it. If constructors with arguments are allowed, builder tools wouldn't know which constructor to call (in case there are many) or what values to pass as parameters.

19:17 / 31:25

Constrained Properties and Customizers and JavaBeans API



VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

THE JAVA BEANS API

- The Java Beans provides a set of classes and interfaces in the **java.beans** package.

INTERFACE	DESCRIPTION
AppletInitializer	Methods in this interface are used to initialize beans that are also applets.
BeanInfo	This interface allows a designer to specify information about the properties, events and methods of a bean.
Customizer	This interface allows a designer to provide a graphical user interface through which a bean may be configured.
DesignMode	Methods in this interface determine if a bean is executing in design mode.

20:47 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

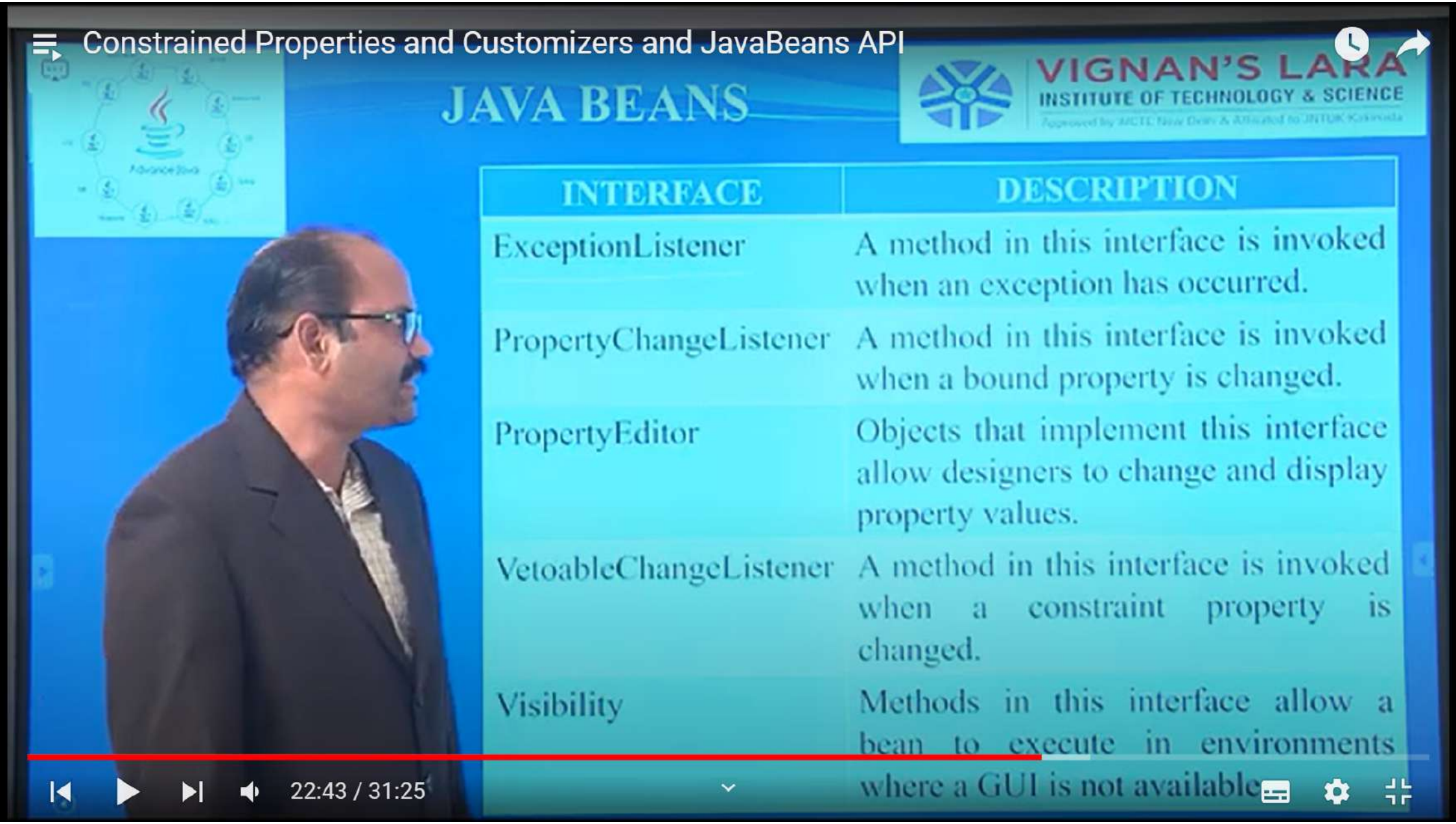
JAVA BEANS

THE JAVA BEANS API

- The Java Beans provides a set of classes and interfaces in the **java.beans** package.

INTERFACE	DESCRIPTION
AppletInitializer	Methods in this interface are used to initialize beans that are also applets.
BeanInfo	This interface allows a designer to specify information about the properties, events and methods of a bean.
Customizer	This interface allows a designer to provide a graphical user interface through which a bean may be configured.
DesignMode	Methods in this interface determine if a bean is executing in design mode.

21:43 / 31:25


Constrained Properties and Customizers and JavaBeans API
VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE
JAVA BEANS
INTERFACE
DESCRIPTION
ExceptionListener
A method in this interface is invoked when an exception has occurred.
PropertyChangeListener
A method in this interface is invoked when a bound property is changed.
PropertyEditor
Objects that implement this interface allow designers to change and display property values.
VetoableChangeListener
A method in this interface is invoked when a constraint property is changed.
Visibility
Methods in this interface allow a bean to execute in environments where a GUI is not available
22:43 / 31:25

Constrained Properties and Customizers and JavaBeans API

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

THE JAVA BEANS API

▶ The following table lists the classes in **java.beans**.

CLASS	DESCRIPTION
BeanDescriptor	This class provides information about a Bean. It also allows to associate a Customizer with a Bean.
Beans	This class is used to obtain information about a Bean.
DefaultPersistenceDelegate	A subclass of PersistenceDelegate.
Encoder	Encodes the state of a set of Beans and can be used to write this information to a stream.

25:03 / 31:25

Constrained Properties and Customizers and JavaBeans API

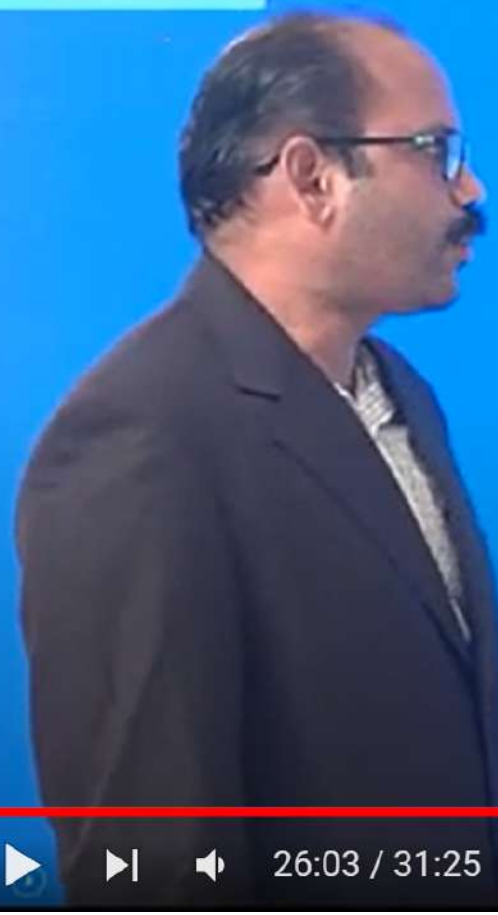
VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

THE JAVA BEANS API

▶ The following table lists the classes in **java.beans**.

CLASS	DESCRIPTION
EventHandler	Supports dynamic event listener creation.
EventSetDescriptor	Instances of this class describe an event that can be generated by a Bean.
Expression	Encapsulates a call to a method that returns a result.



26:03 / 31:25

Constrained Properties and Customizers and JavaBeans API

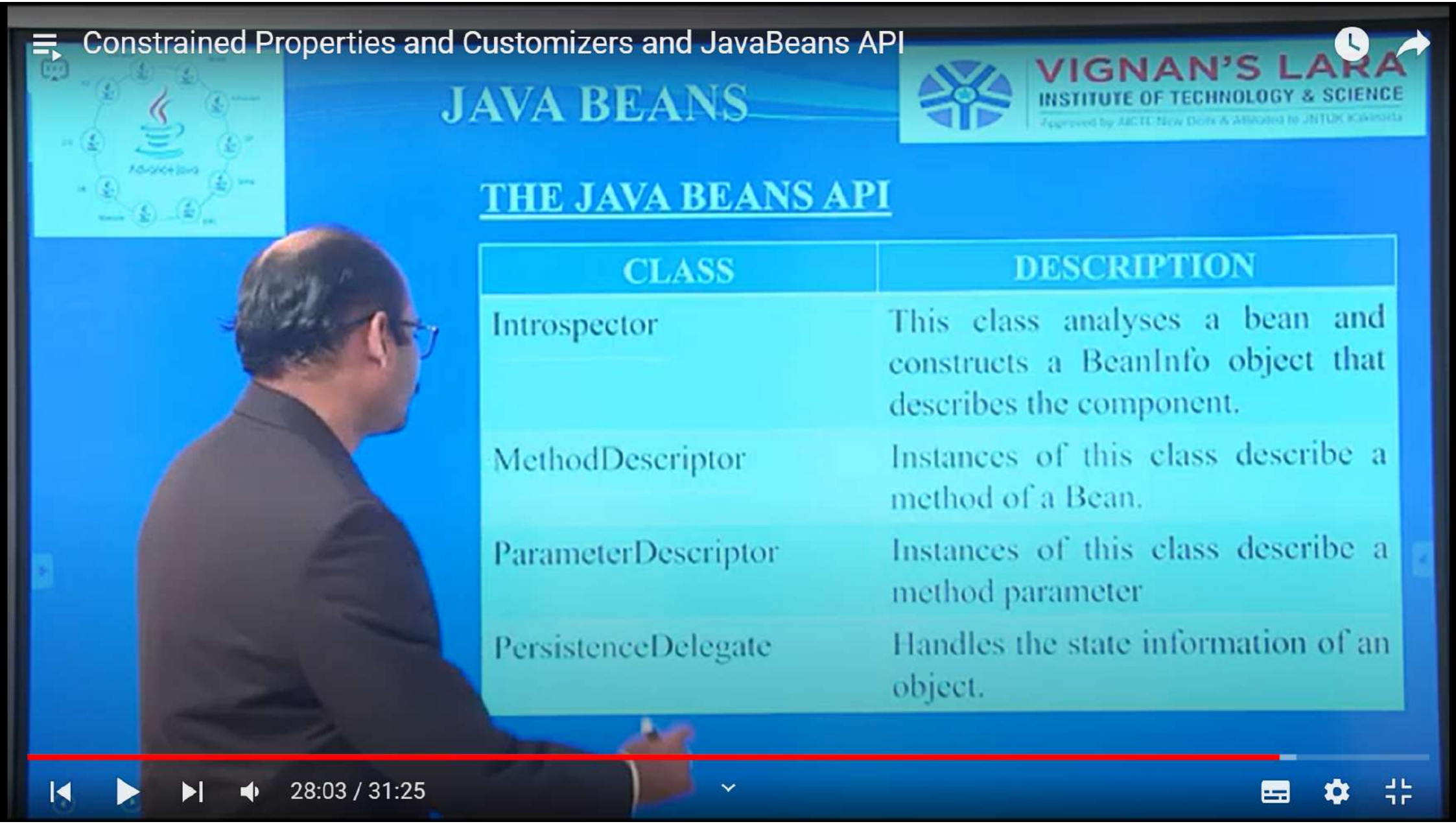
VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE

JAVA BEANS

THE JAVA BEANS API

CLASS	DESCRIPTION
FeatureDescriptor	This is the super class of PropertyDescriptor, EventSetDescriptor and MethodDescriptor classes
IndexedProprtyDescriptor	Instances of this class describe an indexed property of a bean.
InrospectionException	An exception of this type is generated if a problem occurs when analysing a Bean.

27:23 / 31:25


Constrained Properties and Customizers and JavaBeans API
VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE
JAVA BEANS
THE JAVA BEANS API
CLASS
DESCRIPTION
Introspector
This class analyses a bean and constructs a BeanInfo object that describes the component.
MethodDescriptor
Instances of this class describe a method of a Bean.
ParameterDescriptor
Instances of this class describe a method parameter
PersistenceDelegate
Handles the state information of an object.
28:03 / 31:25