Q. 7 What is JSP ? What are the advantages and disadvantages of JSP.

Q. 8 Write a JSP code to print the details entered in the form on the next page. The details are entered in the html page as username in textbox, date of birth in textbox, date of joining in text box, gender in radio button.

Q. 9 What is page directives ? Explain page directives and its attributes.

Q. 10 Write a JSP that swaps two numbers. Input the number through HTML.

Q. 11 Write a JSP application that computes the cubes of the number from 1 to 10, and display in table format.

Q. 12 Explain with an example addition of java bean to a JSP page.

❑❑❑

UNIT - III

CHAPTER 5

# Java Beans

**Syllabus Topics**

**Java Beans :** Introduction, JavaBeans Properties, Examples.

**Syllabus Topic : Introduction**

## 5.1 Introduction

**Q.** What is Java Bean? Discuss the advantages and disadvantages of it.
**Q.** Explain the concept of Java Beans in detail.

☞ **Java Bean :** A Java Bean is a java class. JavaBeans are classes that capture and store many objects into a single object (called bean).

- Let's learn some unique characteristics that distinguish a JavaBean from other Java classes :
  - Java Bean class may have a number of properties which can be used in various ways.
  - Java Bean class may have a number of "**getter**" and "**setter**" methods for the properties which increases user friendliness.
  - Java Bean should be serializable.
  - Java Bean provides a default i.e. no-argument constructor.

☞ **Why use Java Bean ?**

- Java Bean is a reusable software component. It is easy to maintain.
- It encapsulates many objects into one object, so we can access this object from multiple places.

### Advantages of using Java Beans

- Another application can use properties, events, and methods of a bean class.
- A bean may register and/or generate events.
- Other supporting software can be provided to help configure a bean.
- Beans allow us to store configuration settings to persistent storage and restore it.
- Reusability in different environments.
- Used to create applet, servlet, application or other components.
- JavaBeans are dynamic, can be customized.
- Can be deployed in network systems.

### Disadvantages of using Java Beans

- Java bean class is with a no-argument constructor, it may lead to invalid state and developer may not realize the problem.
- JavaBeans are mutable and so definitely it loses the advantages offered by immutable objects.
- Creating getters/setter for almost every property can lead to boilerplate code (sections of code that have to be included in many places with little or no alteration).

**Syllabus Topic : JavaBeans Properties**

## 5.2 JavaBeans Properties

**Q.** Explain Java beans properties in detail.
**Q.** Demonstrate the working of getter and setter methods of JavaBeans class.

- A JavaBean property can be **read, write, read only**, or write only.
- Java beans properties are named properties meaning they have names and meanings that are context-specific.
- JavaBean properties are accessed through following two methods –

| Sr. No. | Method | Description |
|---|---|---|
| 1. | getPropertyName() | Let's consider, if property name is *EName*, your method name would be getEName() to read that property. This method is called accessor. |
| 2. | setPropertyName() | Let's consider, if property name is *EName*, your method name would be setEName() to write that property. This method is called mutator. |

- A write-only attribute will have only a method and a read-only attribute will have only a getPropertyName() method.



### Syntax for setter methods

```
(setPropertyName( ) )
```

1. It should be public in nature.
2. The return-type should be void.
3. The setter method should be prefixed with set.
4. It should take some argument i.e. it should not be no-arg method.

### Syntax for getter methods

```
(getPropertyName())
```

1. It should be public in nature.
2. The return-type should not be void i.e. according to our requirement we have to give return-type.
3. The getter method should be prefixed with get.
4. It should not take any argument.

For Boolean properties getter method name can be prefixed with either "get" or "is". But recommended to use "is".

**Syllabus Topic : JavaBeans Examples**

## 5.3 JavaBeans Examples

### Let's learn a Simple example of java bean class

```
//Book.java

package myPack;
public class Book implements java.io.Serializable
{
   private int Bid;
   private String Bname;

   public Book(){}

   public void setBId(int id)  {Bid=id;}

   public int getBId()  {return Bid;}

   public void setBName(String name)  {Bname=name;}
```

```
    public String getBName()   {return Bname;}
}
```

### How to access the java bean class?

To access the java bean class, we should use getter and setter methods.

```
package myPack;
public class Check
{
    public static void main(String args[])
    {
        Book b=new Book ();
        b.setBName("Java");
        System.out.println(b.getBName());
    }
}
```

### Accessing JavaBeans Properties

- The jsp:useBean action tag is used to locate or instantiate a bean class.
- If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

### Syntax of jsp:useBean action tag

```
<jsp:useBean id= "instanceName" scope= "page | request | session | application"
class= "packageName.className" type= "packageName.className"
beanName="packageName.className | <%= expression >" >
</jsp:useBean>
```

### Attributes and Usage of jsp:useBean action tag

1. **Id :** It uniquely identify the bean in the specified scope.
2. **Scope :** It signifies the scope (page, request, session or application) of the bean. The default scope is page.
   - **Page :** Signifies the scope is within the JSP page. It is a default scope.
   - **Request :** It has wider scope than page. It signifies that the scope is from any JSP page that processes the same request.
   - **Session :** It has wider scope than request. It signifies that the scope is from any JSP page in the same session whether processes the same request or not.
   - **Application :** It has wider scope than session. It signifies that the scope is from any JSP page in the same application.



3. **Class :** It creates an object of the bean class, but it must have no-argument or no constructor. It must not be abstract.
4. **Type :** If the bean already exists in the scope, it provides the bean a data type. It is mainly used with class or beanName attribute. If we use it without class or beanName, no bean is instantiated.
5. **beanName :** Instantiates the bean using the java.beans.Beans.instantiate() method.

We can use the <jsp:getProperty/>action and

<jsp:setProperty/> action with <jsp:useBean...> action

### Syntax

Let's see the syntax –

```
<jsp:useBean id = "id" class = "bean's class" scope = "bean's scope">
<jsp:setProperty name = "bean's id" property = "property name"
 value = "value"/>
<jsp:getProperty name = "bean's id" property = "property name"/>
.....
</jsp:useBean>
```

- The name attribute references the id of a JavaBean previously introduced to the JSP by the useBean action.
- The property attribute is the name of the get or the set methods that should be invoked.

### Example

Following example shows how to access the data using the above syntax –

```
<html>
<head>
    <title>get and set properties Example</title>
</head>
...
<body>
    <jsp:useBean id = "books" class = "myPack.Book">
    <jsp:setProperty name = " books " property = "BId" value = "Android"/>
    <jsp:setProperty name = " books " property = "BName" value = "B904"/>
    </jsp:useBean>

<p>Book Name:
  <jsp:getProperty name = "books" property = "BName"/>
</p>
```

```
    <p>Book Id:
        <jsp:getProperty name = "books" property = "BId"/>
    </p>
</body>
</html>
```

Access the above JSP, the following result will be displayed –

**Output**

```
Book Name : Android
Book Id : B904
```

**Program 5.3.1 :** Design a JavaBean program to illustrate the getName() method on boolean type attribute.

**Soln :**

Basically when we want to deal with Boolean type attribute we do following

```
public class Test
{
  private boolean empty;
  public boolean getName()
  {
    return empty;
  }
  public boolean isempty()
  {
    return empty;
  }
}
```

// Java Program of JavaBean class

```
package MyPack;
public class Student implements java.io.Serializable
{
  private int rno;
  private String sname;
  public Student()
  {
  }
public void setRno(int id)
  {
```



```
    rno= id;
  }
public int getRno()
  {
    return rno;
  }
public void setSname(String name)
  {
    sname = name;
  }
public String getSname()
  {
    return sname;
  }
}
```

// Java program to access JavaBean class

```
package myPack;
public class Test {
public static void main(String args[])
  {
    Student s = new Student(); // object is created
    s.setName("Arvind Muthhu"); // setting value to the object
    System.out.println(s.getName());
  }
}
```

**Output**

```
Arvind Muthhu
```

**Program 5.3.2 :** Design a JavaBean program to store and retrieve different BankAccount details.

**Soln :**

// Java Program of JavaBean class

```
public class BankAccount implements java.io.Serializable
{
  private String accountNumber = null;
```

```
private String totalAmount = null;
private String accountHolderName = null;
private int accountHolderAge = 0;
private String accountHolderAddress = null;

public String getAccountNumber()
{
    return accountNumber;
}
public void setAccountNumber(String accountNumber)
{
    this.accountNumber = accountNumber;
}
public String getTotalAmount()
{
    return totalAmount;
}
public void setTotalAmount(String totalAmount)
{
    this.totalAmount = totalAmount;
}
public String getAccountHolderName()
{
    return accountHolderName;
}
public void setAccountHolderName(String accountHolderName)
{
    this.accountHolderName = accountHolderName;
}
public int getAccountHolderAge()
{
    return accountHolderAge;
}
public void setAccountHolderAge(int accountHolderAge)
{
    this.accountHolderAge = accountHolderAge;
}
public String getAccountHolderAddress()
```



```
{
    return accountHolderAddress;
}
public void setAccountHolderAddress(String accountHolderAddress)
{
    this.accountHolderAddress = accountHolderAddress;
}

}
```

// Java program to access JavaBean class

```
public class MyTest {
public static void main(String args[])
{
    BankAccount b1 = new BankAccount (); // object is created
    b1. setAccountNumber ("A1001"); // setting value to the object
    System.out.println("Account Number is "+b1.getAccountNumber Name());
    b1.setAccountHolderAge(34);
    System.out.println("Age is "+b1.getAccountHolderAge ());

}
}
```

**Output**

```
Account Number is  A1001
Age is 34
```

❑❑❑