



Serialization

Sending Complex Java Data Structures to Files or Over the Network

Originals of Slides and Source Code for Examples:

<http://courses.coreservlets.com/Course-Materials/java.html>

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Agenda

- **Idea**
- **Requirements**
- **Steps for sending data**
- **Steps for receiving data**
- **Example: saving GUI in file**
- **Example: sending GUI across network**



Overview

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Idea of Serialization

- **Java lets you send arbitrarily complex data structures with a single command**
 - writeObject from ObjectOutputStream
 - Can write to file, socket, process, etc.
 - Almost any data type: ArrayList, array, Frame, Panel, custom classes, etc. Arbitrarily nested.
 - Custom classes must implement Serializable
- **Java lets you read complex data structures in a single command**
 - readObject from ObjectInputStream
 - Can read from file, socket, process, etc.
 - Receiver must have class files for custom classes
 - Receiver must be on same major version of Java

Requirements

- **Top-level data structure and all internal components must implement Serializable**
 - Most builtin classes already do
 - ArrayList, HashMap, array, String, Frame/JFrame, Panel/JPanel, Button/JButton, etc.
 - Primitives are OK inside data structures.
 - No need for wrapper classes.
 - Making your own classes serializable is simple
 - Just say “implements Serializable” (no methods!)
 - Bottom-most non-Serializable class must have a zero-argument constructor. (I.e., parent of first Serializable class. Object is OK.)
- **Both ends must use same major version of Java**
 - I.e., sender cannot use 1.6 and receiver use 1.7 or vice versa
- **Both ends must have same version of class files**
 - E.g., if you add a method to your class, old serialized data is no longer valid



Sending Data

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Sending Data: Summary

- **Wrap an ObjectOutputStream around any regular OutputStream**

- To file

```
FileOutputStream fileOut =  
    new FileOutputStream("SomeFile.ser");  
ObjectOutputStream out =  
    new ObjectOutputStream(fileOut);
```

- To socket

```
OutputStream socketOut =  
    someSocket.getOutputStream();  
ObjectOutputStream out =  
    new ObjectOutputStream(socketOut);
```

- **Send top-level data structure**

```
out.writeObject(theData);  
out.close();
```

Sending Data to File: Details (Example for Array of Shapes)

```
try {
    Shape[] shapes = { new Circle(...),
                       new Rectangle(...),
                       ... };
    FileOutputStream fileOut =
        new FileOutputStream("shapes.ser");
    ObjectOutputStream out =
        new ObjectOutputStream(fileOut);
    out.writeObject(shapes);
    out.close();
} catch (IOException ioe) {
    System.out.println("Error sending data" + ioe);
}
```


Sending Data to Socket : Details (Example for Array of Shapes)

```
try {
    Shape[] shapes = { new Circle(...),
                       new Rectangle(...),
                       ... };

    Socket socket = new Socket("host", port);
    // Or Socket socket = serverSock.accept();
    OutputStream socketOut =
        socket.getOutputStream();
    ObjectOutputStream out =
        new ObjectOutputStream(socketOut);
    out.writeObject(shapes);
    out.close();
} catch (IOException ioe) {
    System.out.println("Error sending data" + ioe);
}
```



Receiving Data

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Receiving Data: Summary

- **Wrap an `ObjectInputStream` around any regular `InputStream`**

- From file

```
FileInputStream fileIn =  
    new FileInputStream(new File("SomeFile.ser"));  
ObjectInputStream in =  
    new ObjectInputStream(fileIn);
```

- From socket

```
InputStream socketIn =  
    someSocket.getInputStream();  
ObjectInputStream in =  
    new ObjectInputStream(socketIn);
```

- **Read top-level data structure**

```
SomeType var = (SomeType)in.readObject();
```

Reading Data from File: Details (Example for Array of Shapes)

```
try {
    FileInputStream fileIn =
        new FileInputStream(new File("shapes.ser"));
    ObjectInputStream in =
        new ObjectInputStream(fileIn);
    Shape[] shapes = (Shape[])in.readObject();
} catch (IOException ioe) {
    System.out.println("Error reading file: "
        + ioe);
} catch (ClassNotFoundException cnfe) {
    System.out.println("No such class: " + cnfe);
}
```

Reading Data from Socket: Details (Example for Array of Shapes)

```
try {
    Socket socket = new Socket("host",port);
    // Or Socket socket = serverSock.accept();
    InputStream socketIn = socket.getInputStream();
    ObjectInputStream in
        new ObjectInputStream(socketIn);
    Shape[] shapes = (Shape[])in.readObject();
} catch(IOException ioe) {
    System.out.println("Error reading socket: "
        + ioe);
} catch(ClassNotFoundException cnfe) {
    System.out.println("No such class: " + cnfe);
}
```



Example: Sending Entire Window to File or Network

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Example: SaveableFrame

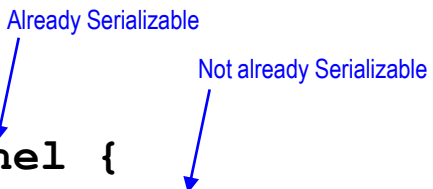
- **Data:**
 - Main Frame (Frame already Serializable)
 - Frame has internal fields (ints) representing width, height, colors, layout manager, and location on screen
 - Two subpanels (Panel already Serializable)
 - Bottom panel has 2 buttons (Button already Serializable)
 - Top panel has:
 - Custom mouse listener that explicitly implements Serializable
 - BetterCircle objects that are created when user presses mouse. (Extends Component, which already implements Serializable)
- **Sending to/from file**
 - Clicking “Save” sends state of Frame to file.
 - If file exists when program starts, old state taken from file
- **Sending to/from network**
 - Server created that sends state of Frame to any client
 - Client created that connects to server and gets copy of Frame

SaveableFrame (Custom Class)

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class CirclePanel extends Panel {
    private class ClickAdapter extends MouseAdapter
        implements Serializable {
        public void mousePressed(MouseEvent event) {
            BetterCircle circle =
                new BetterCircle(Color.BLACK, 25);
            add(circle);
            circle.setCenter(event.getX(), event.getY());
        }
    }

    public CirclePanel() {
        setLayout(null);
        addMouseListener(new ClickAdapter());
    }
}
```



SaveableFrame (Base Code to Send Frame)

- **SaveableFrame.java**

```
public void setFrame(OutputStream rawOut) {
    try {
        ObjectOutputStream out =
            new ObjectOutputStream(rawOut);
        out.writeObject(this);
        out.close();
    } catch(IOException ioe) {
        System.out.println("Error saving frame: " + ioe);
    }
}
```

SaveableFrame (Code to Send Frame to File)

- **SaveableFrame.java**

```
public void actionPerformed(ActionEvent event) {
    if (event.getSource() == clearButton) {
        circlePanel.removeAll();
        circlePanel.repaint();
    } else if (event.getSource() == saveButton) {
        try {
            FileOutputStream fileOut =
                new FileOutputStream("SavedFrame.ser");
            sendFrame(fileOut);
            fileOut.close();
        } catch (IOException ioe) {
            System.out.println("IOException: " + ioe);
        }
    }
}
```

SaveableFrame (Code to Send Frame to Client on Network)

- **FrameServer.java**

```
public void listen(int port, SaveableFrame frame) {
    try {
        ServerSocket listener = new ServerSocket(port);
        Socket server;
        while(true) {
            server = listener.accept();
            frame.sendFrame(server.getOutputStream());
        }
    } catch (IOException ioe) {
        System.out.println("IOException: " + ioe);
        ioe.printStackTrace();
    }
}
```

SaveableFrame (Base Code to Get Frame)

- **SaveableFrame.java**

```
public static SaveableFrame getFrame(InputStream rawIn) {
    SaveableFrame frame = null;
    try {
        ObjectInputStream in = new ObjectInputStream(rawIn);
        frame = (SaveableFrame)in.readObject();
        frame.setVisible(true);
        return(frame);
    } catch(IOException ioe) {
        System.out.println("Error reading file: " + ioe);
    } catch(ClassNotFoundException cnfe) {
        System.out.println("No such class: " + cnfe);
    }
    return(frame);
}
```


SaveableFrame (Code to Get Frame from File)

- **SaveableFrame.java**

```
public static void main(String[] args) {
    SaveableFrame frame;
    File serializeFile = new File(serializeFilename);
    if (serializeFile.exists()) {
        try {
            FileInputStream fileIn =
                new FileInputStream(serializeFile);
            frame = getFrame(fileIn);
        } catch (IOException ioe) {
            System.out.println("IOException: " + ioe);
        }
    } else {
        frame = new SaveableFrame();
    }
}
```

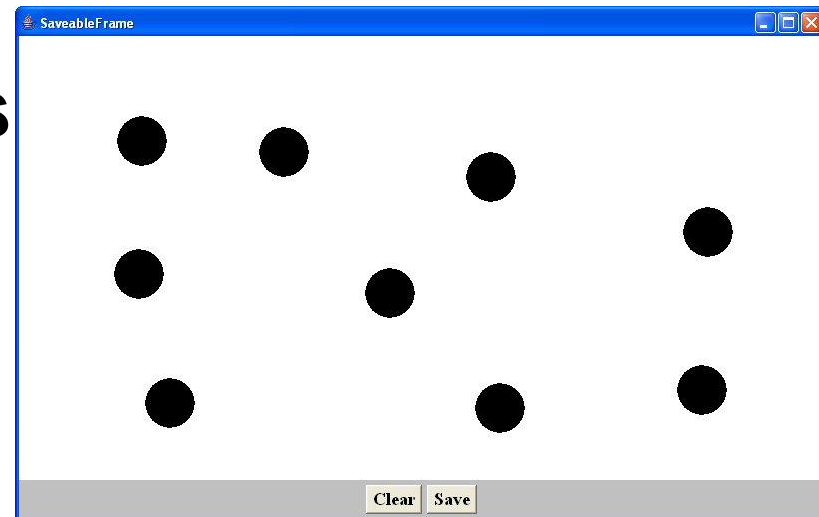
SaveableFrame (Code to Get Frame from Server on Network)

- **FrameClient.java**

```
public FrameClient(String host, int port) {
    try {
        Socket client = new Socket(host, port);
        SaveableFrame frame =
            SaveableFrame.getFrame(client.getInputStream());
    } catch (UnknownHostException uhe) {
        System.out.println("Unknown host: " + host);
        uhe.printStackTrace();
    } catch (IOException ioe) {
        System.out.println("IOException: " + ioe);
        ioe.printStackTrace();
    }
}
```

Results: SaveableFrame (Serialization to/from File)

- **Saving to File**
 - Open frame (600x400, no circles, top left corner)
 - Move window around
 - Resize it
 - Click to add circles
 - Press “Save”
- **Next time program runs**
 - Frame pops up at previous location, with previous size, including previous circles

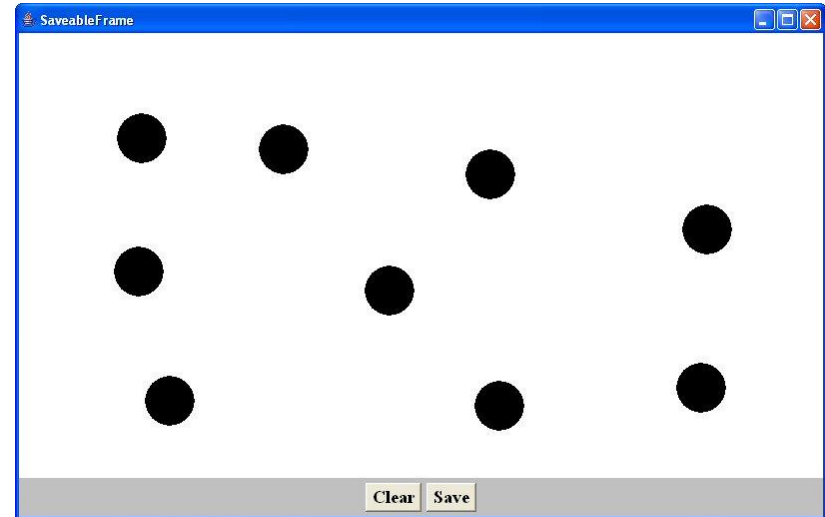


Results: SaveableFrame (Serialization to/from Network)

- **Machine 1**

```
DOS> java FrameServer 8888
```

- Open frame (600x400, no circles, top left corner)
- Move window around
- Resize it
- Click to add circles



- **Machine 2**

```
DOS> java FrameClient coreservlets.com 8888
```

- Frame pops up with same location, size, and circles as version on the server



Wrap-Up

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Summary

- **Class format**

- Make sure custom classes implement Serializable and parent (non-Serializable) class has zero-arg constructors
 - Object is already Serializable

- **Sending data**

- Wrap an ObjectOutputStream around a raw OutputStream
- Call writeObject(objectYouWantToSend)
 - You need to use try/catch blocks

- **Receiving data**

- Wrap an ObjectInputStream around a raw InputStream
- Call readObject
- Cast the result to desired type
 - You need to use try/catch blocks



Questions?

JSF 2, PrimeFaces, Java 7, Ajax, jQuery, Hadoop, RESTful Web Services, Android, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training.

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.