| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SpringLayout.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/Spring.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/SpringLayout.Constraints.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/SpringLayout.html)    [**NO FRAMES**](http://docs.google.com/SpringLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#z337ya) | [METHOD](#1y810tw) |

## **javax.swing**

Class SpringLayout

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.swing.SpringLayout**

**All Implemented Interfaces:** [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html), [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html)

public class **SpringLayout**extends [Object](http://docs.google.com/java/lang/Object.html)implements [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html)

A SpringLayout lays out the children of its associated container according to a set of constraints. See [How to Use SpringLayout](http://java.sun.com/docs/books/tutorial/uiswing/layout/spring.html) in *The Java Tutorial* for examples of using SpringLayout.

Each constraint, represented by a Spring object, controls the vertical or horizontal distance between two component edges. The edges can belong to any child of the container, or to the container itself. For example, the allowable width of a component can be expressed using a constraint that controls the distance between the west (left) and east (right) edges of the component. The allowable *y* coordinates for a component can be expressed by constraining the distance between the north (top) edge of the component and the north edge of its container.

Every child of a SpringLayout-controlled container, as well as the container itself, has exactly one set of constraints associated with it. These constraints are represented by a SpringLayout.Constraints object. By default, SpringLayout creates constraints that make their associated component have the minimum, preferred, and maximum sizes returned by the component's [Component.getMinimumSize()](http://docs.google.com/java/awt/Component.html#getMinimumSize()), [Component.getPreferredSize()](http://docs.google.com/java/awt/Component.html#getPreferredSize()), and [Component.getMaximumSize()](http://docs.google.com/java/awt/Component.html#getMaximumSize()) methods. The *x* and *y* positions are initially not constrained, so that until you constrain them the Component will be positioned at 0,0 relative to the Insets of the parent Container.

You can change a component's constraints in several ways. You can use one of the [putConstraint](http://docs.google.com/javax/swing/SpringLayout.html#putConstraint(java.lang.String,%20java.awt.Component,%20int,%20java.lang.String,%20java.awt.Component)) methods to establish a spring linking the edges of two components within the same container. Or you can get the appropriate SpringLayout.Constraints object using [getConstraints](http://docs.google.com/javax/swing/SpringLayout.html#getConstraints(java.awt.Component)) and then modify one or more of its springs. Or you can get the spring for a particular edge of a component using [getConstraint](http://docs.google.com/javax/swing/SpringLayout.html#getConstraint(java.lang.String,%20java.awt.Component)), and modify it. You can also associate your own SpringLayout.Constraints object with a component by specifying the constraints object when you add the component to its container (using [Container.add(Component, Object)](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20java.lang.Object))).

The Spring object representing each constraint has a minimum, preferred, maximum, and current value. The current value of the spring is somewhere between the minimum and maximum values, according to the formula given in the [Spring.sum(javax.swing.Spring, javax.swing.Spring)](http://docs.google.com/javax/swing/Spring.html#sum(javax.swing.Spring,%20javax.swing.Spring)) method description. When the minimum, preferred, and maximum values are the same, the current value is always equal to them; this inflexible spring is called a *strut*. You can create struts using the factory method [Spring.constant(int)](http://docs.google.com/javax/swing/Spring.html#constant(int)). The Spring class also provides factory methods for creating other kinds of springs, including springs that depend on other springs.

In a SpringLayout, the position of each edge is dependent on the position of just one other edge. If a constraint is subsequently added to create a new binding for an edge, the previous binding is discarded and the edge remains dependent on a single edge. Springs should only be attached between edges of the container and its immediate children; the behavior of the SpringLayout when presented with constraints linking the edges of components from different containers (either internal or external) is undefined.

### SpringLayout vs. Other Layout Managers

**Note:** Unlike many layout managers, SpringLayout doesn't automatically set the location of the components it manages. If you hand-code a GUI that uses SpringLayout, remember to initialize component locations by constraining the west/east and north/south locations.

Depending on the constraints you use, you may also need to set the size of the container explicitly.

Despite the simplicity of SpringLayout, it can emulate the behavior of most other layout managers. For some features, such as the line breaking provided by FlowLayout, you'll need to create a special-purpose subclass of the Spring class.

SpringLayout also provides a way to solve many of the difficult layout problems that cannot be solved by nesting combinations of Boxes. That said, SpringLayout honors the LayoutManager2 contract correctly and so can be nested with other layout managers -- a technique that can be preferable to creating the constraints implied by the other layout managers.

The asymptotic complexity of the layout operation of a SpringLayout is linear in the number of constraints (and/or components).

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

**Since:** 1.4 **See Also:**[Spring](http://docs.google.com/javax/swing/Spring.html), [SpringLayout.Constraints](http://docs.google.com/javax/swing/SpringLayout.Constraints.html)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**SpringLayout.Constraints**](http://docs.google.com/javax/swing/SpringLayout.Constraints.html)            A Constraints object holds the constraints that govern the way a component's size and position change in a container controlled by a SpringLayout. |

| **Field Summary** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**BASELINE**](http://docs.google.com/javax/swing/SpringLayout.html#BASELINE)            Specifies the baseline of a component. |
| static [String](http://docs.google.com/java/lang/String.html) | [**EAST**](http://docs.google.com/javax/swing/SpringLayout.html#EAST)            Specifies the right edge of a component's bounding rectangle. |
| static [String](http://docs.google.com/java/lang/String.html) | [**HEIGHT**](http://docs.google.com/javax/swing/SpringLayout.html#HEIGHT)            Specifies the height of a component's bounding rectangle. |
| static [String](http://docs.google.com/java/lang/String.html) | [**HORIZONTAL\_CENTER**](http://docs.google.com/javax/swing/SpringLayout.html#HORIZONTAL_CENTER)            Specifies the horizontal center of a component's bounding rectangle. |
| static [String](http://docs.google.com/java/lang/String.html) | [**NORTH**](http://docs.google.com/javax/swing/SpringLayout.html#NORTH)            Specifies the top edge of a component's bounding rectangle. |
| static [String](http://docs.google.com/java/lang/String.html) | [**SOUTH**](http://docs.google.com/javax/swing/SpringLayout.html#SOUTH)            Specifies the bottom edge of a component's bounding rectangle. |
| static [String](http://docs.google.com/java/lang/String.html) | [**VERTICAL\_CENTER**](http://docs.google.com/javax/swing/SpringLayout.html#VERTICAL_CENTER)            Specifies the vertical center of a component's bounding rectangle. |
| static [String](http://docs.google.com/java/lang/String.html) | [**WEST**](http://docs.google.com/javax/swing/SpringLayout.html#WEST)            Specifies the left edge of a component's bounding rectangle. |
| static [String](http://docs.google.com/java/lang/String.html) | [**WIDTH**](http://docs.google.com/javax/swing/SpringLayout.html#WIDTH)            Specifies the width of a component's bounding rectangle. |

| **Constructor Summary** | |
| --- | --- |
| [**SpringLayout**](http://docs.google.com/javax/swing/SpringLayout.html#SpringLayout())()            Constructs a new SpringLayout. |

| **Method Summary** | |
| --- | --- |
| void | [**addLayoutComponent**](http://docs.google.com/javax/swing/SpringLayout.html#addLayoutComponent(java.awt.Component,%20java.lang.Object))([Component](http://docs.google.com/java/awt/Component.html) component, [Object](http://docs.google.com/java/lang/Object.html) constraints)            If constraints is an instance of SpringLayout.Constraints, associates the constraints with the specified component. |
| void | [**addLayoutComponent**](http://docs.google.com/javax/swing/SpringLayout.html#addLayoutComponent(java.lang.String,%20java.awt.Component))([String](http://docs.google.com/java/lang/String.html) name, [Component](http://docs.google.com/java/awt/Component.html) c)            Has no effect, since this layout manager does not use a per-component string. |
| [Spring](http://docs.google.com/javax/swing/Spring.html) | [**getConstraint**](http://docs.google.com/javax/swing/SpringLayout.html#getConstraint(java.lang.String,%20java.awt.Component))([String](http://docs.google.com/java/lang/String.html) edgeName, [Component](http://docs.google.com/java/awt/Component.html) c)            Returns the spring controlling the distance between the specified edge of the component and the top or left edge of its parent. |
| [SpringLayout.Constraints](http://docs.google.com/javax/swing/SpringLayout.Constraints.html) | [**getConstraints**](http://docs.google.com/javax/swing/SpringLayout.html#getConstraints(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) c)            Returns the constraints for the specified component. |
| float | [**getLayoutAlignmentX**](http://docs.google.com/javax/swing/SpringLayout.html#getLayoutAlignmentX(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) p)            Returns 0.5f (centered). |
| float | [**getLayoutAlignmentY**](http://docs.google.com/javax/swing/SpringLayout.html#getLayoutAlignmentY(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) p)            Returns 0.5f (centered). |
| void | [**invalidateLayout**](http://docs.google.com/javax/swing/SpringLayout.html#invalidateLayout(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) p)            Invalidates the layout, indicating that if the layout manager has cached information it should be discarded. |
| void | [**layoutContainer**](http://docs.google.com/javax/swing/SpringLayout.html#layoutContainer(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Lays out the specified container. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**maximumLayoutSize**](http://docs.google.com/javax/swing/SpringLayout.html#maximumLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Calculates the maximum size dimensions for the specified container, given the components it contains. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**minimumLayoutSize**](http://docs.google.com/javax/swing/SpringLayout.html#minimumLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Calculates the minimum size dimensions for the specified container, given the components it contains. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**preferredLayoutSize**](http://docs.google.com/javax/swing/SpringLayout.html#preferredLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Calculates the preferred size dimensions for the specified container, given the components it contains. |
| void | [**putConstraint**](http://docs.google.com/javax/swing/SpringLayout.html#putConstraint(java.lang.String,%20java.awt.Component,%20int,%20java.lang.String,%20java.awt.Component))([String](http://docs.google.com/java/lang/String.html) e1, [Component](http://docs.google.com/java/awt/Component.html) c1, int pad, [String](http://docs.google.com/java/lang/String.html) e2, [Component](http://docs.google.com/java/awt/Component.html) c2)            Links edge e1 of component c1 to edge e2 of component c2, with a fixed distance between the edges. |
| void | [**putConstraint**](http://docs.google.com/javax/swing/SpringLayout.html#putConstraint(java.lang.String,%20java.awt.Component,%20javax.swing.Spring,%20java.lang.String,%20java.awt.Component))([String](http://docs.google.com/java/lang/String.html) e1, [Component](http://docs.google.com/java/awt/Component.html) c1, [Spring](http://docs.google.com/javax/swing/Spring.html) s, [String](http://docs.google.com/java/lang/String.html) e2, [Component](http://docs.google.com/java/awt/Component.html) c2)            Links edge e1 of component c1 to edge e2 of component c2. |
| void | [**removeLayoutComponent**](http://docs.google.com/javax/swing/SpringLayout.html#removeLayoutComponent(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) c)            Removes the constraints associated with the specified component. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### NORTH

public static final [String](http://docs.google.com/java/lang/String.html) **NORTH**

Specifies the top edge of a component's bounding rectangle.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.NORTH)

### SOUTH

public static final [String](http://docs.google.com/java/lang/String.html) **SOUTH**

Specifies the bottom edge of a component's bounding rectangle.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.SOUTH)

### EAST

public static final [String](http://docs.google.com/java/lang/String.html) **EAST**

Specifies the right edge of a component's bounding rectangle.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.EAST)

### WEST

public static final [String](http://docs.google.com/java/lang/String.html) **WEST**

Specifies the left edge of a component's bounding rectangle.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.WEST)

### HORIZONTAL\_CENTER

public static final [String](http://docs.google.com/java/lang/String.html) **HORIZONTAL\_CENTER**

Specifies the horizontal center of a component's bounding rectangle.

**Since:** 1.6 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.HORIZONTAL_CENTER)

### VERTICAL\_CENTER

public static final [String](http://docs.google.com/java/lang/String.html) **VERTICAL\_CENTER**

Specifies the vertical center of a component's bounding rectangle.

**Since:** 1.6 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.VERTICAL_CENTER)

### BASELINE

public static final [String](http://docs.google.com/java/lang/String.html) **BASELINE**

Specifies the baseline of a component.

**Since:** 1.6 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.BASELINE)

### WIDTH

public static final [String](http://docs.google.com/java/lang/String.html) **WIDTH**

Specifies the width of a component's bounding rectangle.

**Since:** 1.6 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.WIDTH)

### HEIGHT

public static final [String](http://docs.google.com/java/lang/String.html) **HEIGHT**

Specifies the height of a component's bounding rectangle.

**Since:** 1.6 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.SpringLayout.HEIGHT)

| **Constructor Detail** |
| --- |

### SpringLayout

public **SpringLayout**()

Constructs a new SpringLayout.

| **Method Detail** |
| --- |

### addLayoutComponent

public void **addLayoutComponent**([String](http://docs.google.com/java/lang/String.html) name,  
 [Component](http://docs.google.com/java/awt/Component.html) c)

Has no effect, since this layout manager does not use a per-component string.

**Specified by:**[addLayoutComponent](http://docs.google.com/java/awt/LayoutManager.html#addLayoutComponent(java.lang.String,%20java.awt.Component)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**name - the string to be associated with the componentc - the component to be added

### removeLayoutComponent

public void **removeLayoutComponent**([Component](http://docs.google.com/java/awt/Component.html) c)

Removes the constraints associated with the specified component.

**Specified by:**[removeLayoutComponent](http://docs.google.com/java/awt/LayoutManager.html#removeLayoutComponent(java.awt.Component)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**c - the component being removed from the container

### minimumLayoutSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **minimumLayoutSize**([Container](http://docs.google.com/java/awt/Container.html) parent)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)) Calculates the minimum size dimensions for the specified container, given the components it contains.

**Specified by:**[minimumLayoutSize](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**parent - the component to be laid out**See Also:**[LayoutManager.preferredLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container))

### preferredLayoutSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **preferredLayoutSize**([Container](http://docs.google.com/java/awt/Container.html) parent)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container)) Calculates the preferred size dimensions for the specified container, given the components it contains.

**Specified by:**[preferredLayoutSize](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**parent - the container to be laid out**See Also:**[LayoutManager.minimumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container))

### maximumLayoutSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **maximumLayoutSize**([Container](http://docs.google.com/java/awt/Container.html) parent)

**Description copied from interface:** [**LayoutManager2**](http://docs.google.com/java/awt/LayoutManager2.html#maximumLayoutSize(java.awt.Container)) Calculates the maximum size dimensions for the specified container, given the components it contains.

**Specified by:**[maximumLayoutSize](http://docs.google.com/java/awt/LayoutManager2.html#maximumLayoutSize(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html) **See Also:**[Component.getMaximumSize()](http://docs.google.com/java/awt/Component.html#getMaximumSize()), [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html)

### addLayoutComponent

public void **addLayoutComponent**([Component](http://docs.google.com/java/awt/Component.html) component,  
 [Object](http://docs.google.com/java/lang/Object.html) constraints)

If constraints is an instance of SpringLayout.Constraints, associates the constraints with the specified component.

**Specified by:**[addLayoutComponent](http://docs.google.com/java/awt/LayoutManager2.html#addLayoutComponent(java.awt.Component,%20java.lang.Object)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html) **Parameters:**component - the component being addedconstraints - the component's constraints**See Also:**[SpringLayout.Constraints](http://docs.google.com/javax/swing/SpringLayout.Constraints.html)

### getLayoutAlignmentX

public float **getLayoutAlignmentX**([Container](http://docs.google.com/java/awt/Container.html) p)

Returns 0.5f (centered).

**Specified by:**[getLayoutAlignmentX](http://docs.google.com/java/awt/LayoutManager2.html#getLayoutAlignmentX(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html)

### getLayoutAlignmentY

public float **getLayoutAlignmentY**([Container](http://docs.google.com/java/awt/Container.html) p)

Returns 0.5f (centered).

**Specified by:**[getLayoutAlignmentY](http://docs.google.com/java/awt/LayoutManager2.html#getLayoutAlignmentY(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html)

### invalidateLayout

public void **invalidateLayout**([Container](http://docs.google.com/java/awt/Container.html) p)

**Description copied from interface:** [**LayoutManager2**](http://docs.google.com/java/awt/LayoutManager2.html#invalidateLayout(java.awt.Container)) Invalidates the layout, indicating that if the layout manager has cached information it should be discarded.

**Specified by:**[invalidateLayout](http://docs.google.com/java/awt/LayoutManager2.html#invalidateLayout(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html)

### putConstraint

public void **putConstraint**([String](http://docs.google.com/java/lang/String.html) e1,  
 [Component](http://docs.google.com/java/awt/Component.html) c1,  
 int pad,  
 [String](http://docs.google.com/java/lang/String.html) e2,  
 [Component](http://docs.google.com/java/awt/Component.html) c2)

Links edge e1 of component c1 to edge e2 of component c2, with a fixed distance between the edges. This constraint will cause the assignment

value(e1, c1) = value(e2, c2) + pad

to take place during all subsequent layout operations.

**Parameters:**e1 - the edge of the dependentc1 - the component of the dependentpad - the fixed distance between dependent and anchore2 - the edge of the anchorc2 - the component of the anchor**See Also:**[putConstraint(String, Component, Spring, String, Component)](http://docs.google.com/javax/swing/SpringLayout.html#putConstraint(java.lang.String,%20java.awt.Component,%20javax.swing.Spring,%20java.lang.String,%20java.awt.Component))

### putConstraint

public void **putConstraint**([String](http://docs.google.com/java/lang/String.html) e1,  
 [Component](http://docs.google.com/java/awt/Component.html) c1,  
 [Spring](http://docs.google.com/javax/swing/Spring.html) s,  
 [String](http://docs.google.com/java/lang/String.html) e2,  
 [Component](http://docs.google.com/java/awt/Component.html) c2)

Links edge e1 of component c1 to edge e2 of component c2. As edge (e2, c2) changes value, edge (e1, c1) will be calculated by taking the (spring) sum of (e2, c2) and s. Each edge must have one of the following values: SpringLayout.NORTH, SpringLayout.SOUTH, SpringLayout.EAST, SpringLayout.WEST, SpringLayout.VERTICAL\_CENTER, SpringLayout.HORIZONTAL\_CENTER or SpringLayout.BASELINE.

**Parameters:**e1 - the edge of the dependentc1 - the component of the dependents - the spring linking dependent and anchore2 - the edge of the anchorc2 - the component of the anchor**See Also:**[putConstraint(String, Component, int, String, Component)](http://docs.google.com/javax/swing/SpringLayout.html#putConstraint(java.lang.String,%20java.awt.Component,%20int,%20java.lang.String,%20java.awt.Component)), [NORTH](http://docs.google.com/javax/swing/SpringLayout.html#NORTH), [SOUTH](http://docs.google.com/javax/swing/SpringLayout.html#SOUTH), [EAST](http://docs.google.com/javax/swing/SpringLayout.html#EAST), [WEST](http://docs.google.com/javax/swing/SpringLayout.html#WEST), [VERTICAL\_CENTER](http://docs.google.com/javax/swing/SpringLayout.html#VERTICAL_CENTER), [HORIZONTAL\_CENTER](http://docs.google.com/javax/swing/SpringLayout.html#HORIZONTAL_CENTER), [BASELINE](http://docs.google.com/javax/swing/SpringLayout.html#BASELINE)

### getConstraints

public [SpringLayout.Constraints](http://docs.google.com/javax/swing/SpringLayout.Constraints.html) **getConstraints**([Component](http://docs.google.com/java/awt/Component.html) c)

Returns the constraints for the specified component. Note that, unlike the GridBagLayout getConstraints method, this method does not clone constraints. If no constraints have been associated with this component, this method returns a default constraints object positioned at 0,0 relative to the parent's Insets and its width/height constrained to the minimum, maximum, and preferred sizes of the component. The size characteristics are not frozen at the time this method is called; instead this method returns a constraints object whose characteristics track the characteristics of the component as they change.

**Parameters:**c - the component whose constraints will be returned **Returns:**the constraints for the specified component

### getConstraint

public [Spring](http://docs.google.com/javax/swing/Spring.html) **getConstraint**([String](http://docs.google.com/java/lang/String.html) edgeName,  
 [Component](http://docs.google.com/java/awt/Component.html) c)

Returns the spring controlling the distance between the specified edge of the component and the top or left edge of its parent. This method, instead of returning the current binding for the edge, returns a proxy that tracks the characteristics of the edge even if the edge is subsequently rebound. Proxies are intended to be used in builder envonments where it is useful to allow the user to define the constraints for a layout in any order. Proxies do, however, provide the means to create cyclic dependencies amongst the constraints of a layout. Such cycles are detected internally by SpringLayout so that the layout operation always terminates.

**Parameters:**edgeName - must be one of SpringLayout.NORTH, SpringLayout.SOUTH, SpringLayout.EAST, SpringLayout.WEST, SpringLayout.VERTICAL\_CENTER, SpringLayout.HORIZONTAL\_CENTER or SpringLayout.BASELINEc - the component whose edge spring is desired **Returns:**a proxy for the spring controlling the distance between the specified edge and the top or left edge of its parent**See Also:**[NORTH](http://docs.google.com/javax/swing/SpringLayout.html#NORTH), [SOUTH](http://docs.google.com/javax/swing/SpringLayout.html#SOUTH), [EAST](http://docs.google.com/javax/swing/SpringLayout.html#EAST), [WEST](http://docs.google.com/javax/swing/SpringLayout.html#WEST), [VERTICAL\_CENTER](http://docs.google.com/javax/swing/SpringLayout.html#VERTICAL_CENTER), [HORIZONTAL\_CENTER](http://docs.google.com/javax/swing/SpringLayout.html#HORIZONTAL_CENTER), [BASELINE](http://docs.google.com/javax/swing/SpringLayout.html#BASELINE)

### layoutContainer

public void **layoutContainer**([Container](http://docs.google.com/java/awt/Container.html) parent)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#layoutContainer(java.awt.Container)) Lays out the specified container.

**Specified by:**[layoutContainer](http://docs.google.com/java/awt/LayoutManager.html#layoutContainer(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**parent - the container to be laid out

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SpringLayout.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/Spring.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/SpringLayout.Constraints.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/SpringLayout.html)    [**NO FRAMES**](http://docs.google.com/SpringLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#z337ya) | [METHOD](#1y810tw) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).