

## Stored Override Example

This page provides a quick run-through of what the `stored_override` example does. For more details, you should refer to the source, which you will find in the download kit under:

```
examples/stored_override
```

You can build and run the example by simply `cd`'ing into that directory and typing `ant`.

### Example Scenario

Suppose that we have a simple JSR175 annotation type called `DeploymentInfo` which specifies a 'cacheSize' attribute for some kind of deployment unit:

#### DeploymentInfo.java

```
@AnnogenInfo(  
    annoBeanClass =  
    "org.codehaus.annogen.examples.stored_override.DeploymentInfoAnnoBean"  
)  
@Retention(RetentionPolicy.RUNTIME)  
@Target(ElementType.TYPE)  
  
public @interface DeploymentInfo {  
    public int cacheSize();  
  
}
```

Suppose that we also have an deployment unit called `TinyCacheEJB` which has a `DeploymentInfo` annotation:

## TinyCacheEJB.java

```
@DeploymentInfo(cacheSize = 3)
public class TinyCacheEJB {
    //...
}
```

This example demonstrates a very simple way in which we might use Annogen to allow for the value of `cacheSize` to be overridden.

### Generate Phase

The first thing we have to do is generate an `AnnoBean` that will act as a proxy to instances of `DeploymentInfo`. To do this, we simply run Annogen's compilation task on `DeploymentInfo`:

## build.xml

```
<annogen outputdir='${codegen_dir}'
         classpathref='classpath'
         srcdir='src'

         includes='org/codehaus/annogen/examples/stored_override/DeploymentInfo.java'
         implementAnnotationTypes='true' />
```

This generates a `DeploymentInfoAnnoBean` that implements `DeploymentInfo`:

## DeploymentInfoAnnoBean.java

```
public class DeploymentInfoAnnoBean extends
AnnoBeanBase implements DeploymentInfo {

    private int cacheSize;

    public int cacheSize() {
        return this.cacheSize;
    }

    public void set_cacheSize(int in) {
        this.cacheSize = in;
    }
}
```

### Override Phase

Notice that the generated `DeploymentInfoAnnoBean` class has a setter for `cacheSize`. We're now going to create an [AnnoOverride](#) which uses that setter to effectively change the size of the cache on `TinyCacheEJB`. We create an instance of [StoredAnnoOverride](#), which is a helper class Annogen provides for building simple `AnnoOverriders`.

## User.java

```
StoredAnnoOverride storedOverrides =
StoredAnnoOverride.Factory.create();
```

`StoredAnnoOverride` requires us to make an [ElementId](#) for the `TinyCacheEJB` class:

```
ReflectElementIdPool elementPool =  
ReflectElementIdPool.Factory.create();  
ElementId tinyCacheEJBClassId =  
elementPool.getIdFor(TinyCacheEJB.class);
```

...which we will use to get an instance of the `DeploymentInfoAnnoBean` which applies to `TinyCacheEJB`.

```
AnnoBeanSet annos =  
storedOverrides.findOrCreateStoredAnnoSetFor  
(tinyCacheEJBClassId);  
DeploymentInfoAnnoBean annoBean =  
(DeploymentInfoAnnoBean)  
  
annos.findOrCreateBeanFor(DeploymentInfo.class);
```

The `StoredAnnoOverrider` will provide this `DeploymentInfoAnnoBean` as a replacement for the `DeploymentInfo` annotation on `TinyCacheEJB`. So, if we change the value of `cacheSize` on it:

```
annoBean.set_cacheSize(newCacheSize);
```

We have changed the value of cache size for `TinyCacheEJB` as it will appear when our framework code views its `DeploymentInfo` annotation.

## View Phase

Compared to the `Override` phase, the `View` phase is very simple. All we have to do is create a [ReflectAnnoViewer](#) which knows about the `AnnoOverrider` we created above:

## DeployerTool.java

```
annoViewer =  
ReflectAnnoViewer.Factory.create(myAnnoOverr  
ider);
```

This `annoViewer` can now tell us about any annotations we want to know about, and it will take our overrides into account. So, if we get the `DeploymentInfo` for `TinyCacheEJB`:

```
DeploymentInfo deploymentInfo =  
(DeploymentInfo)  
  
annoViewer.getAnnotation(DeploymentInfo.class,  
TinyCacheEJB.class);
```

we will get back a `DeploymentInfo` annotation object which reflects our `cacheSize` override.

In reality, `deploymentInfo` is an instance of `DeploymentInfoAnnoBean`, but one nice thing here is that it is an instance of our regular 175 annotation type, `DeploymentInfo`. We can look at the `cacheSize` value on the `TinyCacheEJB` and just go deploy it without a bunch of extra messy logic to check whether the value was overridden.

This illustrates one of the key advantages that Annogen provides: the logic for overriding annotations is cleanly partitioned away from the code which is trying to do something with those annotations.