# Power Objects: The Object Finder

Many applications have the need to locate some resource, or object during initialization or when perform their assigned tasks. The process of locating this resource can be as simple as choosing between two classes, or as onerous as producing an object based on a hint of information.

A solution to this problem arrives in the form of the Object Finder framework from JWorkbench. This simple framework is comprised of three classes: IFinder, ObjectNotFoundException, and ObjectFinder.

The IFinder interface specifies the functionality to be implemented by an object that will procure other objects, finding them locally or remotely, in order to satisfy an application's request. The first method attempts to find an object given an object key. Typically this is done if the certainty of finding the object is great, or guaranteed. The second method attempts to find an object given an object key, but is always passed a default object key (which is normally a guaranteed find) to ensure that at least one object can be found.

**IFinder:**

```
public interface IFinder {

	public Object find(String objectKey);

	public Object find(String objectKey, String defaultObjectKey);
}
```

Next we have the simple exception class that provides for run time notification of object procurement issues using the Exception chaining available in jdk1.4.

**ObjectNotFoundException:**

```
public class ObjectNotFoundException extends RuntimeException {

	public ObjectNotFoundException() {
		super();
	}
	public ObjectNotFoundException(Throwable cause) {
		super(cause);
	}
}
```

Lastly, and most importantly, we have the ObjectFinder class, a default implementation of IFinder. This class implements the single argument find method by attempting locate a class name in the Java system properties and produce an object. This simple method can easily be used to load a default class into an application with the ability to substitute another class using the Java VM's –D option.

**ObjectFinder:**

```
public class ObjectFinder implements IFinder {

	public ObjectFinder() {
		super();
	}

	public Object find(String objectKey) {
		Object object = null;

		String className = System.getProperty(objectKey);
		if (className != null) {
			try {
				object = Class.forName(className)
						.newInstance();
			}
			catch (Exception exception) {
				throw new ObjectNotFoundException(exception);
			}
		}
		return object;
	}

	public Object find(String objectKey, String defaultObjectKey) {

		Object object = find(objectKey);
		if (object == null) {
			try {
				object = Class.forName(defaultObjectKey)
						.newInstance();
			}
			catch (Exception exception) {
				throw new ObjectNotFoundException(exception);
			}
		}
		return object;
	}
}
```

Whether used in an API implementation to encourage programmers to supply or supplant some internal mechanism, application testing, or as a means of dynamic class loading, the ObjectFinder presents a simple easy to use mechanism to assist programmers in making their application or API more flexible.

The Mjolnir Exception framework uses the ObjectFinder to help developer's replace the framework's default ExceptionManager!