WebObjects with Scala

What is Scala?

Scala is a modern, multi-paradigm JVM language that is most often compared to Groovy, Clojure and Erlang. Its functional language foundations and built-in Actors library make it especially attractive for concurrent computing. (Scala is an abbreviation for "scalable" hinting at its design goals).

In this day and age of multi-core processors concurrent computing can not be ignored. Many of the design features of Scala have been chosen with concurrency in mind, some of which may not be unfamiliar to Objective-C or WebObjects developers. Here's a summary:

	Objective-C	**Java**	**Scala**
Immutability	Partial - via collections *e.g: NSArray/NSMutableArray*	No	Yes
Closures	Yes - via Blocks (*Extension*)	No	Yes - via Anonymous Functions
Static variables	Yes	Yes	No
Static methods	Yes	Yes	No
Concurrency	Yes - via Grand Central Dispatch (*Extension*)	*Yes - via Threads*	Yes - via Actors
	Weakly Typed	~~**Strongly Typed**~~	**Strongly Typed**

Other notable features include:

	Objective-C	**Java**	**Scala**
Parametered methods	Yes *e.g: addObject: to:*	No	Yes *e.g: add(object= ,to=)*
Class composition	Yes - via Categories	Yes - via Interfaces	Yes - via Traits

Why Use Scala?

For WebObjects developers, Scala offers itself as a powerful, safe and easy-to-use solution for concurrent computing. (In other words, Scala Actors can be used for problems that would have normally required threads).

Can WebObjects be Programmed In Scala?

Yes. It is very simple.
By virtue of being a JVM-language, Scala compiles to java bytecode.

Furthermore, being a multi-paradigm language grants Scala easy WebObjects-interoperability.

Caveats

Legacy tool support is often cited as a weak point. The Eclipse Scala plugin has been found to be slow at times and sometimes buggy.

WebObjects In Scala

The following highlights some of the differences between Java and Scala in WebObjects:

EOs in Scala

Thread-Safe Shared Vars

Scala doesn't have static variables or methods. Instead Scala employs the Singleton Pattern which is built into the language and is **thread-safe**: a class can have a *Companion Object* that will allow you to achieve something equivalent to static variables - but better.

You don't have to worry about synchronizing access to shared mutable fields in a concurrent application.
(This is not however true when for example you have a `val` declared as a `NSMutableArray`. You will still have to synchronize when adding to or removing from this mutable field).

The following is an example of the use of a *Companion Object* for Talent in Scala instead of Talent static fields in Java.

Java:

```
public class Talent extends EOGenericRecord {
        public static final String ENTITY_NAME        = "Talent";
}
```

Scala:

```
object Talent {
        val ENTITY_NAME = "Talent"
}
```

This value will be accessed exactly the same way in both languages:

```
Talent.ENTITY_NAME
```

Compacted imports

In Java:

```
import com.webobjects.eocontrol.EOGenericRecord;
import com.webobjects.eocontrol.EORelationshipManipulation;
```

In Scala:

```
import com.webobjects.eocontrol.{EOGenericRecord, EORelationshipManipulation}
```

WOComponents in Scala

Compact Constructors

Scala allows for simpler use of multi-valued constructors than Java.

In Java:

```
public class MenuHeader extends WOComponent {

    public MenuHeader(WOContext aContext) {
        super(aContext);
    }
}
```

In Scala:

```
class MenuHeader(context: WOContext) extends WOComponent(context: WOContext)
```

Simplified Exception Handling

Scala doesn't force you to catch exceptions unlike in Java.
In addition, the syntax employs Scala's very powerful **pattern matching** to handle exceptions.

In Java:

```
try {
    EditPageInterface epi = D2W.factory().editPageForNewObjectWithEntityNamed(_manipulatedEntityName,
session());
    epi.setNextPage(context().page());
    nextPage = (WOComponent) epi;
} catch (IllegalArgumentException e) {
    ErrorPageInterface epf = D2W.factory().errorPage(session());
    epf.setMessage(e.toString());
    epf.setNextPage(context().page());
    nextPage = (WOComponent) epf;
}
```

In Scala:

```
try {
     var epi: EditPageInterface = D2W.factory.editPageForNewObjectWithEntityNamed(_manipulatedEntityName,
session)
     epi.setNextPage(context.page)
     nextPage = epi.asInstanceOf[WOComponent]
} catch {
     case e: IllegalArgumentException => {
            var epf: ErrorPageInterface = D2W.factory.errorPage(session)
            epf.setMessage(e.toString)
            epf.setNextPage(context.page)
            nextPage = epf.asInstanceOf[WOComponent]
     }
}
```

Scala Annotations vs. Generated Accessors

Here's an example of accessing variables in the following languages:

	Objective-C	Java	Scala
getter	`[WO:object name]`	`object.name()`	`object.name`
setter	`[object setName:aName]`	`object.setName(aName)`	`object.name = aName`

Of course in Java, we may generate WebObjects classes with "get" methods as well in order to stick to convention.
In scala there is an additional convenience we may use to produce "get" and "set" methods in addition to the default Scala accessors - Scala Annotations.

E.g, in Main.scala we annotate our component keys with `@BeanProperty` to automatically create public "set" and "get" methods.
These variables can then be accessed via *KVC*.

```
import scala.reflect.BeanProperty

@BeanProperty var username = new String()
@BeanProperty var password = new String()
@BeanProperty var isAssistantCheckboxVisible = false
```

How to Use Scala Collections with EOF

To use the Scala Collections API with an NSArray or NSDictionary you simply need to add an import:

```
import scala.collection.JavaConversions._
```

After that, you may access the typical Scala collection methods directly on NSArray. This employs a feature of Scala known as implicit conversions to automagically cast a NSArray (a Java Iterable) into a Scala Iterable while leaving the actual object unchanged.

How to Add Scala to a WO Project (in Eclipse)

1. Install the Scala eclipse IDE
2. Add Scala support to your WO project:
    a. Right-click your project in the WO Explorer
    b. In the context menu select Configure -> Add Scala Nature
3. Convert to or use `.scala` instead of `.java` source

WO Scala Example

The following example is an almost 100% Scala WO app. In reality it is a mixed Java/Scala app:
All the EO logic and WO components are in Scala.
Only the Application class remains Java.

It is based on the D2W Movies example.

File	Modified
ZIP Archive WOScalaExample.zip Proof of concept example v.6	Aug 10, 2010 by Ravi Mendis

Setup

1. Install the Scala eclipse IDE
2. Right-click on Application.java and run as a WOApplication (as usual).

Application can be made into a Scala class as well, but then you will have to create a launcher in Eclipse manually.

EO Templates

When you create your `.eogen` file, be sure to make the following changes in the EOGenerator Editor:

1. Point to the local Scala versions of the .eotemplate files for `Entity` and `_Entity`
2. Change the File Names Extension to "scala"
3. In Destination Paths set the Superclass Package (e.g: base)
4. Uncheck Java under Options

How to Build & Deploy a WebObjects Scala Project with Ant

1. Download and install Scala
2. Set `scala.home` (the location Scala has been installed onto) in the project `build.properties` file
3. Add the scalac task and properties to the ant build.xml file
4. Run from the project directory: `sudo ant clean install`