Scripting for Testers
Lab Instructions for Web Services Lab

The Timeclock Web Services Interface *defines the different messages that the server accepts. You've seen several of them demonstrated already.*

The exact nature of this lab will depend on how far we've gotten in the class, how we've deviated from the plan, how the hardware's behaving, etc.

Getting Started

In the first part of the lab, we'll interact with the server through irb.

1. You'll work in the `C:\timeclock\tutorial` folder. That was the folder you created when you installed all the class materials.
2. Start IRB in the normal way.

You now have a choice. You can run both the server and the client on your local machine. Or your client can talk to a server on an instructor machine.

Choice 1: Starting a local client and server

1. Load up utility functions:
   ```
   irb> require 'timeclock-web-services'
   ```
2. Start a session (for user "web-services-default-user"):
   ```
   irb> session = start_session()
   ```

Choice 2: Starting a client to a remote server

1. You'll need to get a file named `remote-web-services.rb` from the class's shared disk. Put it in the `C:\timeclock\tutorial` folder.
2. Load up utility functions:
   ```
   irb> require 'remote-web-services'
   ```
3. If we're using the default instructor machine, you can skip this step. If not, we'll give you an IP address, something like "192.168.0.23". You'll need to identify that machine to the utilities in the file you required:
   ```
   irb> pick_host("192.169.0.23")
   ```
4. Start a session for a user of your choice. (Try to avoid a name other people are already using.)
   ```
   irb> session = start_session_for("my unique user")
   ```

Run wild!

The *Timeclock Web Services Interface* document defines a number of timeclock commands. They should seem familiar because most of them straightforwardly correspond to the browser interface. Try them out. For example, here's a transcript of how one of us used them:

Copyright © 2003 by Brian Marick and Bret Pettichord. All rights reserved.

```
irb(main):001:0> require 'remote-web-services'
true
irb(main):002:0> pick_host("192.168.0.17")
"192.168.0.17"
irb(main):003:0> session = start_session_for('dr. dawn')
#<Timeclock::RichlyCallingWrapper:0x358210
@wrapped=#<DRb::DRbObject:0x357860
@uri="druby://192.168.0.172:21961", @ref=1420178>>
irb(main):004:0> session.jobs
{}
irb(main):005:0> clinics = Job.named('clinics')
(Job clinics [] {})
irb(main):006:0> session.accept_job(clinics)
true
irb(main):007:0> session.jobs
{"clinics"=>(Job clinics [] {})}
irb(main):008:0> session.start('clinics', Time.now)
(Job clinics [] {})
irb(main):009:0> session.records
[(ActiveRecord 0: clinics
1067185638/0/running/pot_volunteer=false)]
irb(main):010:0> session.stop('clinics', Time.now)
<FinishedRecord 1: clinics for 32.022702 from Sun Oct 26
10:27:18 CST 2003>
```

Try lots of different things. Ask questions when you want to do something but don't know how. We may be able to reveal new functions that do what you want.

Automated Tests

At some point in the class, we may have shown you the testing framework `Test::Unit`. You can find some tests that use it in the file `web-test.rb`. You can run those tests from the command prompt like this:

```
C:\timeclock\tutorial> ruby web-test.rb
```

Spend some time automating some of the web services tests you ran manually earlier. Ask questions about anything you don't understand - the whole class will benefit.