GradebookWS:
A Brief Manual

CS 491 Spring 2015

Group Members:

Sari Sabouh (Scrum Master)
Cody Hilyer
Jonathan Roberts
Alix Rosarion

Table of Contents

Introduction

This guide has been created as a means of helping anyone with invested interests to use the Gradebook web services offered by Blackboard. In truth, very little of this guide is original, but we decided that it would be helpful for all of the setup process – from beginning to end – to be available and thoroughly explained in one resource.

Of course, the first question one might ask is, "Why go through all of this trouble just to use Blackboard web service methods?" Well…

As powerful as Blackboard's web services are, you will need to do a bit of work before you can access their functionality, which has been a daunting task for everyone who worked on this project so far. However, luckily for the reader, this guide exists to simplify the setup

process, and is (hopefully) clear enough to help avoid the many twists and turns our team (and many others) took over the course of researching this.

The Basic Tools

Since the goal of this guide is to make Blackboard's web services accessible to *anyone*, we will start from scratch by first setting up Eclipse Indigo and installing the Maven plugin for it, and then using the Rampart and Axis libraries. All of these are steps you ***must*** take in order to use the project which will be introduced in the next section. It should be noted that this is written under the assumption that the Java Developer's Kit is installed, and the appropriate environment variables (e.g., PATH for the location of the Java "bin" folder) have been set up already.

Installing Eclipse Indigo

1) Go to the download page for Eclipse Indigo by clicking this sentence. Once there, choose the version appropriate to your operating system and download it to whichever folder you prefer.

   Note: Our team has not used the versions for Mac OS X or Linux, so if you experience problems with setup at any time in these operating systems, this guide may not be able to help you. Sorry!

2) Extract the archive to a directory of your choice. If you wish to put Eclipse somewhere like C:\Program Files, we **strongly recommend** extracting to a location which doesn't require administrator privileges first, *then* moving the extracted Eclipse folder to your desired directory. (For example, I chose the directory C:\Program Files\Eclipse.) If you have difficulties doing this despite following this recommendation, try using **7zip**, whose

download page can be accessed by clicking here. Because 7zip is more robust than other archive extraction applications, it may at least help you identify the cause of the problem.

3) Now, we need to make sure Eclipse knows the directory of your java development kit installation. Find the file in Eclipse's main folder named **eclipse.ini** and open it in some text-editing software **with administrator privileges**. (Notepad worked just fine for us). It may look like a mess, but that's okay! Look for **--launcher.defaultAction** in this file, and press enter/return to split into lines which look like the following (adding information as necessary):

```
--launcher.defaultAction
openFile
-vm
C:/Program Files/Java/jdk1.7.0_40/bin/javaw.exe
```

(Replace the last line with your specific javaw directory path.)

Last but not least, save the file. If it fails to save, you probably didn't open it with administrator privileges, so go back and do that.

4) To conclude this section, open Eclipse. The icon should look like an indigo-colored sphere with three lines going through the middle.

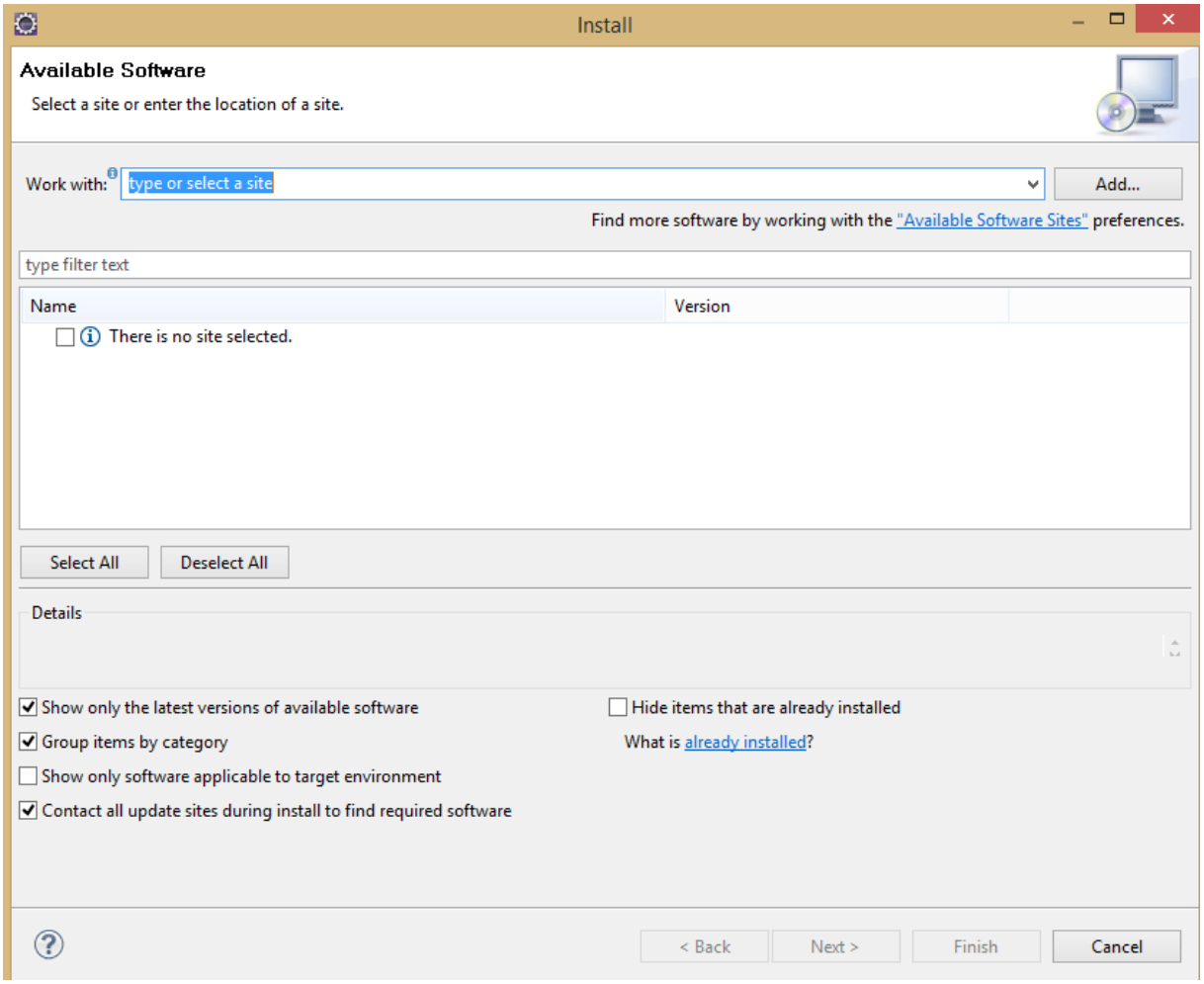
Maven Setup

1) Download the most recent version of Maven from this site.
2) Following the same process as in the Eclipse installation, unzip the Maven archive and place it in a directory of your choice. **Remember this directory for step 4!**
   (I chose C:\Program Files\Apache Software Foundation\apache-maven-3.2.5)
3) Open up the environment variables window. If you forgot how to find environment variables, open System settings, click on **Advanced System Settings**, and then click on **Environment Variables**. (Pro-tip: **Windows key + Pause/Break** will open up System settings immediately.)
4) Create a **New** user variable (or system variable) called **M2_HOME**. Set its variable value equal to the directory you chose in part 2.
5) Create another **New** user variable (or system variable) called **M2**. Set its variable value **exactly** equal to the following: **%M2_HOME%\bin**
6) Create one more **New** user variable (or system variable) called **JAVA_HOME**. This will simply have a variable value equal to the directory of your Java development kit installation. (For example, mine is C:\Program Files\Java\jdk1.7.0_40)

Installing the Maven Plugin for Eclipse

1) Once Eclipse is opened, click the **Help** menu and select **Install New Software…**

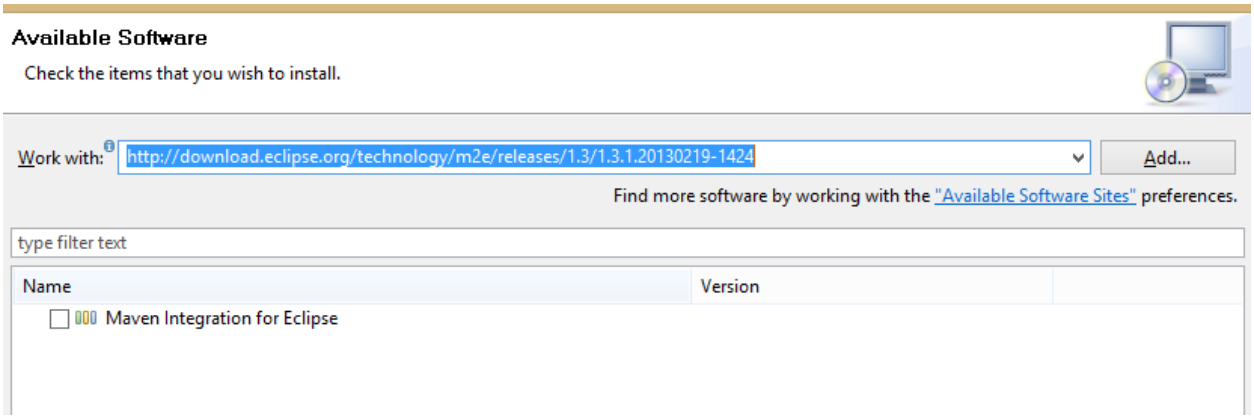
   You should see a prompt like this:



2) In the **Work with** field, place the following URL:

   http://download.eclipse.org/technology/m2e/releases/1.3/1.3.1.20130219-1424

   After clicking **Add…**, the prompt should change and look like the following:



3) Click the checkbox next to "Maven Integration for Eclipse" and then click **Next** at the bottom. Continue until the plugin installation wizard is finished.

Axis Setup

1) Open up the Axis2 downloads page by clicking this sentence. Download the latest stable release (currently 1.6.2 for us) as a zip archive.
2) As you can probably guess, extract the zip archive to somewhere convenient, using tips from the Eclipse installation instructions in case of administrator privilege problems. **Remember this directory you choose for step 4**.
3) Upen up the environment variables window.
4) Create a **New** user (or system) variable called **AXIS2_HOME**. Set its variable value equal to the path chosen in step 2. (I used C:\Program Files\axis2-1.6.2)

Rampart Setup

1) Open up the Rampart downloads page by clicking this sentence. Download the most recent stable release. (Ours is currently 1.6.2).
2) Extract the zip archive to somewhere convenient, using tips from the Eclipse installation instructions in case of administrator privilege problems. **Remember this directory for setting up the bbws Properties file in the next section.**

Using the Proxy Registration Tool

Before continuing, now would be a good time to inform the reader that access to an administrator account on Blackboard is absolutely crucial. Without it, there is no way a proxy tool can even be authorized.

Who is Bruce Phillips?

Bruce Phillips is a Java Architect at the University of Kansas, and is the main reason this guide can even be written. All you really need to know is this:

Bruce wrote several tutorials in his blog about how to access Blackboard web services, along with download links for the updates he made to his project that follow along with the topic of each tutorial. They're really well-written, so if you'd prefer to just follow those, here is a set of links:

Part 1 – Log in with the proxy tool

Part 2 – Web service classes and methods

Part 3 – Generating and using web service client classes

Part 4 – Do many things with user accounts (e.g. creation, deletion, getting)

Opening the Project in Eclipse

1) Download Bruce Phillips' BlackboardCoursesForUser project by clicking this sentence.
2) Extract the zip archive wherever.
3) Open Eclipse.
4) From the menu, navigate to File => Import. You should see a prompt asking you to select an import source.
5) Scroll down until you find a folder named Maven in the prompt. Click it, then select "**Existing Maven Projects**."
6) Click **Browse…** and find the extracted project in the directory you chose in part 2.
7) In the Projects field, click the checkbox next to the only project listed.
8) Click **Finish** to complete the process of importing to your workbench.

Important Requirement 1: The bbws Properties File

Upon opening the project, you will see many folders. Navigate (while in the Eclipse IDE) to **blackboardcoursesforuser\src\main\resources**. Next, open up the file that is named **bbws.properties**. The following fields are left blank, for you to fill in accordingly:

- bbws.blackboardServerURL: Enter the IP address or URL for the Blackboard login page which you use for administrator login
- bbws.clientVendorId: Enter any name you want, it really makes no difference.
- bbws.clientProgramId: Once again, choose a name; it doesn't matter which.
- bbws.modulePath: Remember that directory from Rampart setup step 2? That goes here, but remember that \ is an escape character, so you need to double them.
  (e.g. `C:\\Program Files\\Rampart`)
- bbws.toolRegistrationPassword: This is a password you can find by logging into Blackboard with an administrator account. Go to the **System Admin** tab (top right), then navigate to **Building Blocks** and click on **Proxy Tools**. Finally, click **Manage Global Properties** to view the required proxy tool password.
- bbws.sharedSecret: This is another password you can find by logging into Blackboard with an administrator account. Go to the **System Admin** tab (top right), then navigate to **Web Services** and click the gray **Manage Web Services** button to see your shared secret. (For us, it is currently akl390g2kdalb3)
- bbws.toolDescription: This is yet another field whose value is up to you.
- bbws.username: This can be left blank until the next section. However, if this bothers you, simply put a valid username in this field.

Important Requirement 2: The Application Context File

In the same directory as the previous important requirement (**…\src\main\resources**), open up the applicationContext_BBWS.xml file. Make sure you click the **Source** tab below and left of the code-viewing window to see it as actual text. Scroll down to where you see the following code (which should start at line 27):

```
<property name="toolMethods">

<list>

<value>Context.WS:emulateUser</value>
```

This continues with several more values in a list. The important thing to know here is that each one of these values is listed in a "contract" of sorts when the proxy tool registers itself on Blackboard. This is important because **if you want to access a method later which was not listed here upon registration, it'll be impossible**. This is a mild nuisance because **the only way to modify said "contract" is to delete a proxy tool and re-register with the new applicationContext file**.

Luckily, I have all of the methods which we plan to use from the Gradebook web service listed below, complete with XML tags:

```
<value>Gradebook.WS:getServerVersion</value>
<value>Gradebook.WS:initializeGradebookWS</value>
<value>Gradebook.WS:getRequiredEntitlements</value>
<value>Gradebook.WS:getGradebookColumns</value>
<value>Gradebook.WS:saveGrades</value>
<value>Gradebook.WS:saveColumns</value>
<value>Gradebook.WS:deleteColumns</value>
<value>Gradebook.WS:getGrades</value>
<value>Gradebook.WS:saveAttempts</value>
<value>Gradebook.WS:deleteAttempts</value>
<value>Gradebook.WS:getAttempts</value>
<value>Gradebook.WS:saveGradebookTypes</value>
<value>Gradebook.WS:deleteGradebookTypes</value>
```

```
<value>Gradebook.WS:getGradebookTypes</value>
<value>Gradebook.WS:deleteGrades</value>
<value>Gradebook.WS:updateColumnAttribute</value>
<value>Gradebook.WS:saveGradingSchemas</value>
<value>Gradebook.WS:getGradingSchemas</value>
<value>Gradebook.WS:deleteGradingSchemas</value>
```

Simply make sure all of the above are in the applicationContext_BBWS.xml file immediately after the line containing: `<value>Course.WS:getCourse</value>`.

Running the Proxy Registration Tool

First of all, I want to emphasize that **if you didn't read the 2 previous sub-sections, this will not work as intended**. If you worked through them and have the correct values in the bbws.properties file, simply navigate (in Eclipse) to the following (ridiculously long) directory:

`src\main\java\edu\ku\it\si\registerproxytool\app`

Right click on RegisterProxyToolApp.java (which should be the only file there) and click **Run as…**, then select **Java Application**. If successful, you should see two lines in the Eclipse Console: The first should contain the word "DEBUG" and the second should contain the word "INFO" and let you know that you have successfully registered the proxy tool.

Now log into Blackboard with your administrator account. Go to the **System Admin** tab (top right), then navigate to **Building Blocks** and click on **Proxy Tools**. There should be a new proxy tool listed there, with the same name as the clientProgramId field from the bbws.properties file. While hovering over your proxy tool's row in the **Program** column, a gray circle with a downward-pointing chevron should appear. Click that gray circle, and on the popup menu choose the **Edit** option. Under the Availability section, select **Yes** to permit use of the proxy tool, and finally click **Submit** to save changes. Your proxy tool is now ready to use!

Accessing Gradebook Web Services

Making the GradebookWS Wsdl File

Log into Blackboard with your administrator account. Go to the **System Admin** tab, then navigate to **Web Services**. Look for the **Gradebook.WS** row, and click the URL in the **WSDL Location** column. Right-click the webpage this brings up, and select view source (or whatever your browser equivalent is).

Now, make a new file in the `src\main\resources` directory while in Eclipse. Name the file **gradebookWS.wsdl**. Finally, copy and paste the source code from the website mentioned above into the file, and save it.

Generating a Stub File with Axis

While in Eclipse, open the file named AxisCodeGenerator.java, which is located in the directory `src\main\java\edu\ku\it\si\blackboardcoursesforuser\axis`. Now, change lines 26 and 28 as follows:

26: Change this line to read:

```
"-p", "edu.ku.it.si.bbgradebookws.generated",
```

28: Change this line to read:

```
"-uri", "src/main/resources/gradebookWS.wsdl" });
```

Now, right-click on the AxisCodeGenerator.java file in the left explorer pane, and select **Run as…** followed by **Java Application**.

You are now ready to call methods from the GradebookWS API!

Methods We Created

Anomalies with Blackboard's Web Service API

IDs:

When giving Blackboard an ID string (required by various method calls), the string cannot be blank. However, it *can* be an incorrect value; the server will then modify it to be correct. The format for this ID string is `_XXX_X` where each X represents some digit (0-9).

setDescriptionType (String descriptionType) :

The descriptionType parameter passed in when calling this method *must* be "HTML" with no exceptions, even though the API states that "PLAIN_TEXT" and "SMART_TEXT" are possibilities.

API

```
void changeGrade(ScoreVO scoreVO, String grade)
```

Changes the value of the grade attribute in the ScoreVO object to the value specified by the string grade.

```
ScoreVO[] generateGradableAttempts(ScoreVO[] scores,
                                   ColumnVO[] columnsLackingAttempts)
```

Checks all columns in the passed-in parameter ColumnVO array which do not have an attempt, and puts in attempts to save the trouble of putting them in with the Blackboard interface. (This allows a professor to grade an assignment which did not have a submission on Blackboard.) The ScoreVO array returned represents the updated collection of ScoreVO objects.

```
ColumnVO[] createColumn(ColumnVO[] col, String name)
```

This creates a column (gradable item) as an alternative to making it with the Blackboard interface. The name parameter specifies the name of the column (assignment, test, etc.)

```
void deleteColumn(ColumnVO[] gradebookColumns, int index)
```

Given an array of ColumnVO objects (which represent gradebook columns, like assignments and tests – basically gradable stuff), this will delete one column within that array, which is specified by the index passed in.

```
String displayGrades(List<String> gradeList)
```

Organizes the passed-in parameter gradeList into a neat string object which displays them in an organized fashion.

```
void publishColumns(SaveColumns save)
```

This updates the server with locally changed column information. Only called after a column is modified or created. The SaveColumns object is ready for this method by first creating it normally with a constructor, then using the method setColumns and setCourseId

```
void publishGrades(ScoreVO[] grades, String courseId)
```

Updates the grades (contained in the array of ScoreVO objects) for a particular course (specified by the courseId string) that have been changed locally on the Blackboard server.

Where to Go from Here

Issues Encountered

We were not able to access the Id object (for an object of Course, Student, etc.) because it comes directly from the Blackboard site, which cannot be directly accessed as far as we know. This is doable, but time constraints stopped us from figuring it out.

The main purpose of accessing the Id object would be the ability to list all the student names in a course, which would connect with the algorithm that we wrote for abbreviating student names while maintaining uniqueness.

Building Off Our Work

It is a great idea to gain accessibility to more web services. There are **many** available and thus, there is a great deal of functionality that we have not even looked at.

Furthermore, refactoring the code we wrote for greater ability to build would be a good plan. The majority of what we did with this was research on *how* to get things working, and did not pay nearly enough attention to style.

After this, a GUI would definitely be a great route to take, so that instructors with a wide range of skill with computer literacy could make use of it.

Another idea would be to tie together Dr. Thornton's Leaderboard project with this ease of Web Service accessibility, so that Leaderboard can become a much more powerful program with fewer dependencies on Blackboard.