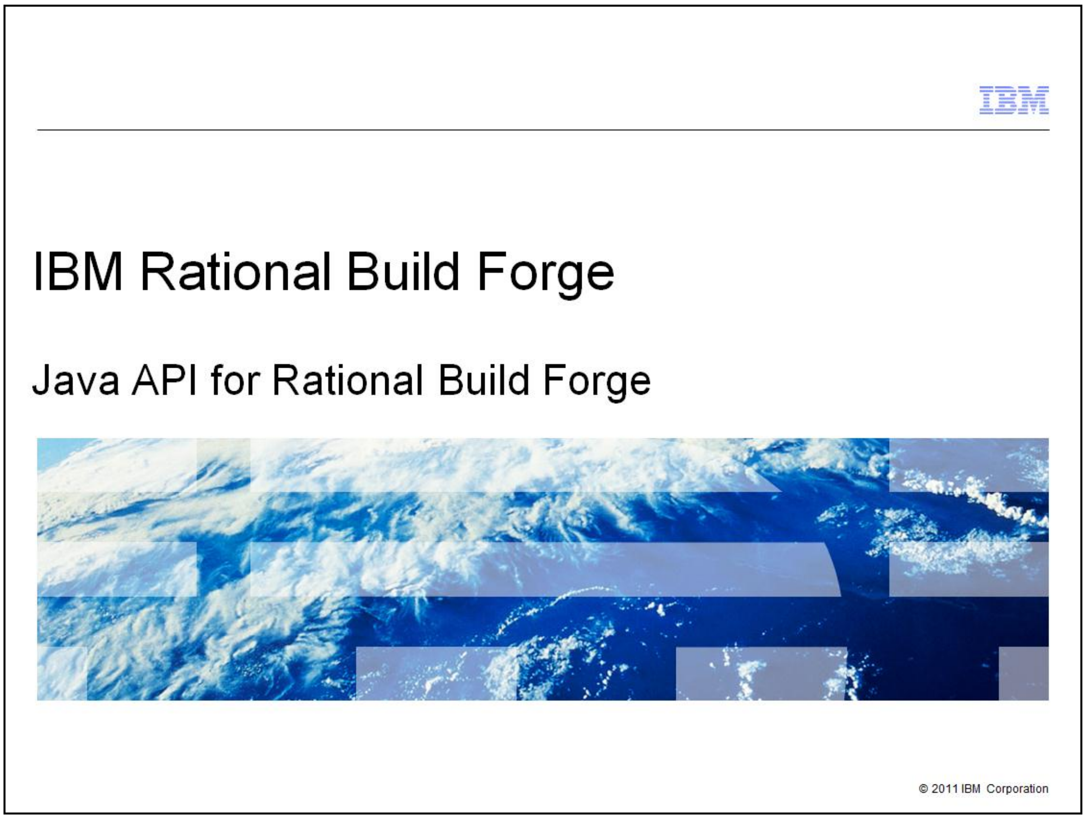


IBM Rational Build Forge

Java API for Rational Build Forge

© 2011 IBM Corporation

This is a presentation of the Java™ API for IBM Rational® Build Forge®.

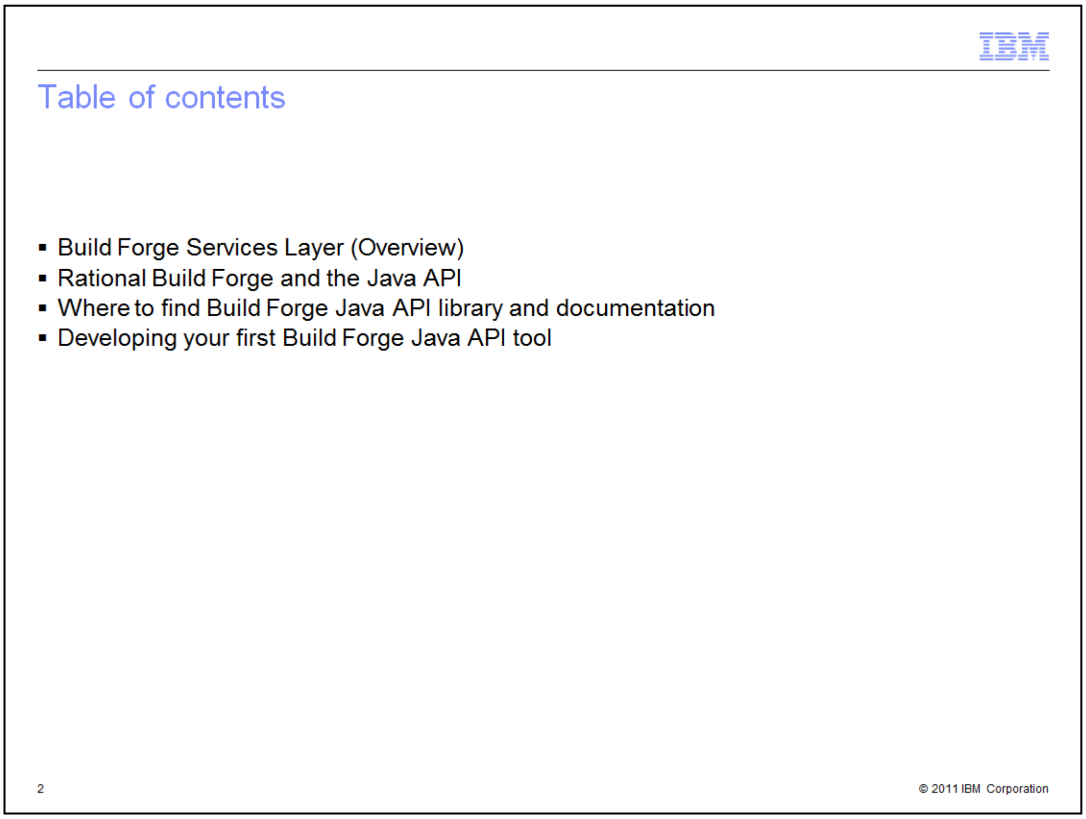


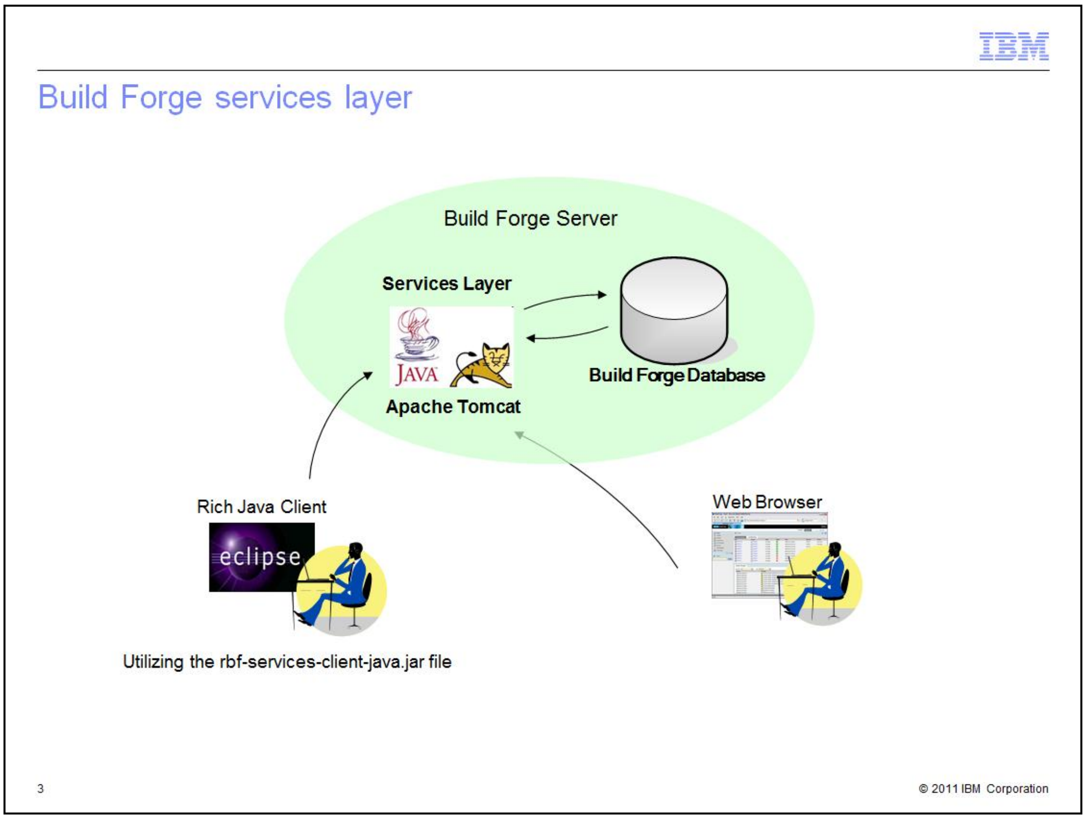
Table of contents

2

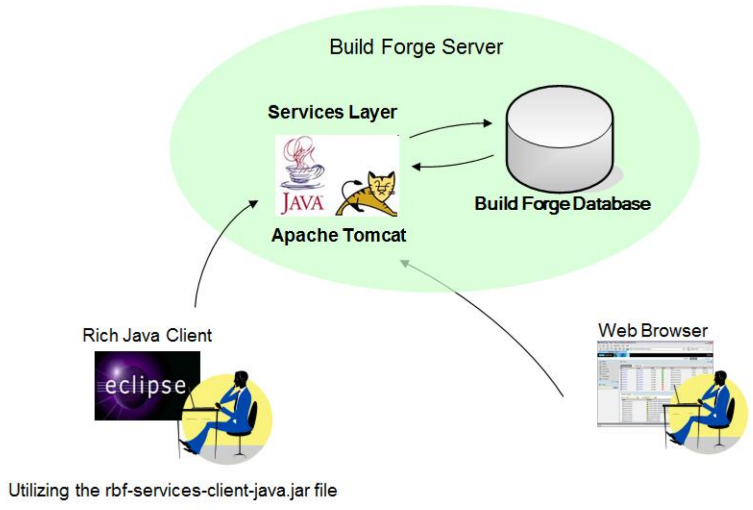
© 2011 IBM Corporation

This presentation will include an overview of the Build Forge Services Layer. It will cover the Java based API for Rational Build Forge. There is information on where to find the Build Forge Java API library and its documentation.

This presentation will also cover the development of your Build Forge Java API tool. Presentation of the development process includes the setup of the Java project. It also covers including the rbf-services-client-java.jar file as a Build Library and the importing the JavaDoc file rbf-services-client-java-docs.zip for reference.



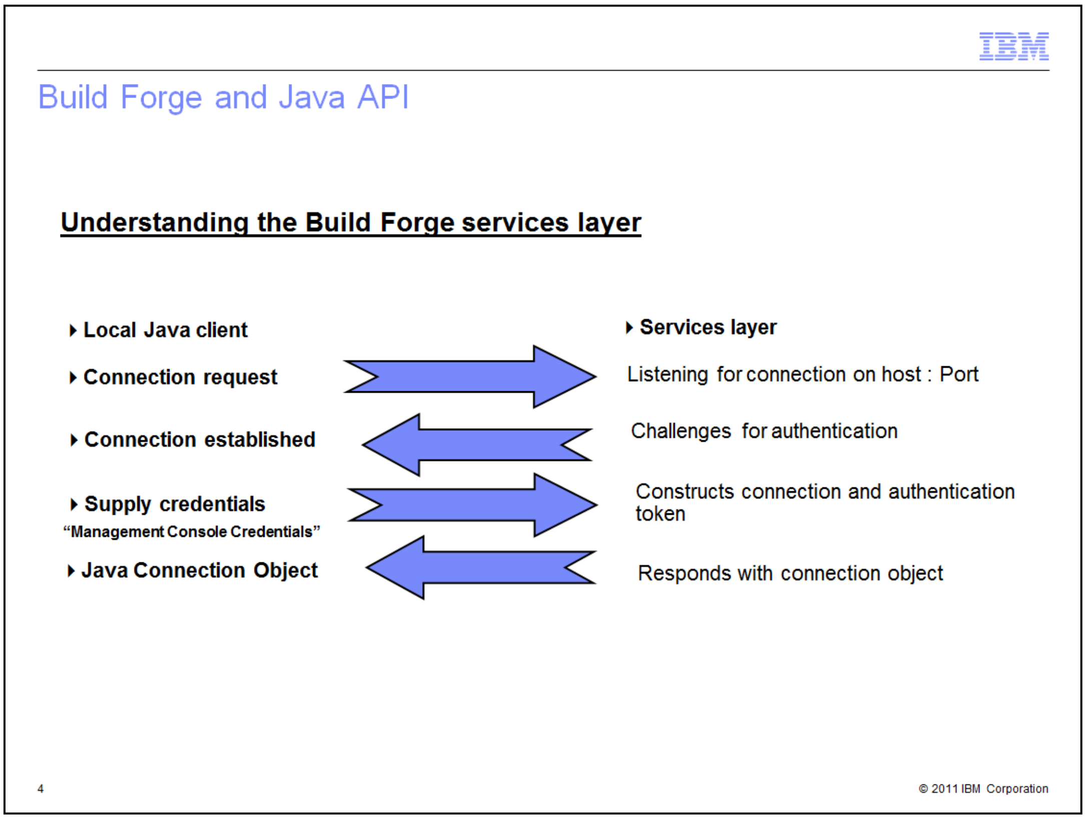
Build Forge services layer



3

© 2011 IBM Corporation

This is a diagram of the Java Services Layer architecture. It is deployed as a web application in Apache Tomcat or IBM WebSphere®. The web application establishes a direct connection to the Rational Build Forge database. The web application listens on designated SSL and non-SSL TCP ports. Web access occurs using HTTP or HTTPS. Thick client access occurs using the bf-services-client-java.jar file.



Build Forge and Java API

Understanding the Build Forge services layer

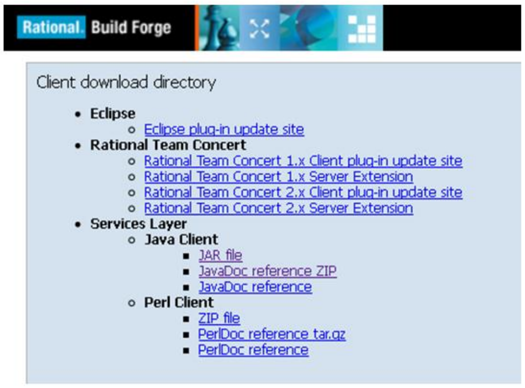
Local Java client
Services layer
Connection request
Listening for connection on host : Port
Connection established
Challenges for authentication
Supply credentials
"Management Console Credentials"
Constructs connection and authentication token
Java Connection Object
Responds with connection object

© 2011 IBM Corporation

Here is an overview of the Build Forge Services Layer. The Local Java Client makes a Service Layer API call to connect on the identified port. The Services Layer TCP connection is established. Local Java Client makes an authentication call to the Services Layer. The Services Layer then returns a complete APIClientConnection object containing connection details and the authentication token.

The Local Client application can now reuse this object for the remainder of the Tools execution or until the timeout period is reached.

Build Forge API library and documentation



Rational. Build Forge

Client download directory

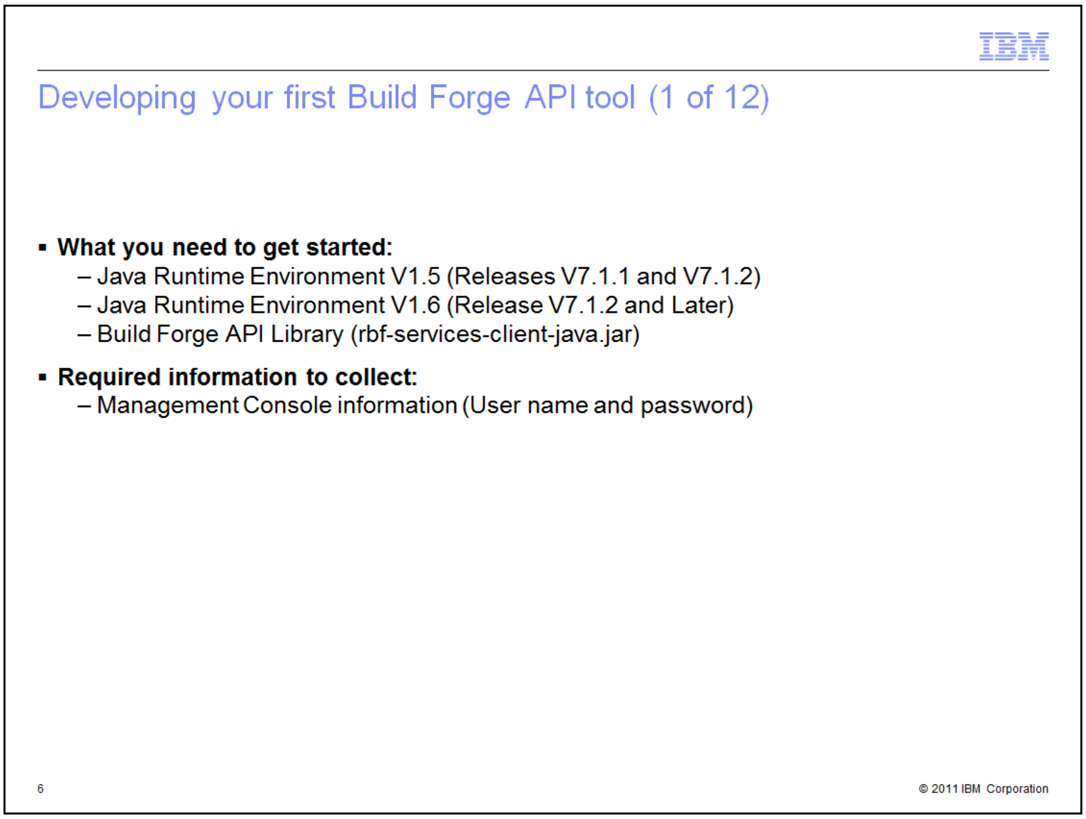
- **Eclipse**
  - Eclipse plug-in update site
- **Rational Team Concert**
  - Rational Team Concert 1.x Client plug-in update site
  - Rational Team Concert 1.x Server Extension
  - Rational Team Concert 2.x Client plug-in update site
  - Rational Team Concert 2.x Server Extension
- **Services Layer**
  - **Java Client**
    - JAR file
    - JavaDoc reference ZIP
    - JavaDoc reference
  - **Perl Client**
    - ZIP file
    - PerlDoc reference tar.gz
    - PerlDoc reference

Access URL:

http://<Build Forge Server>:<Port>/clients

© 2011 IBM Corporation

This is an example Client download directory page. Log into the Management Console to be redirected to the Client download directory. From here, you can download both rbf-services-client-java.jar (a .jar file) and rbf-services-client-java-docs.zip (a .zip archive containing the JavaDoc reference).



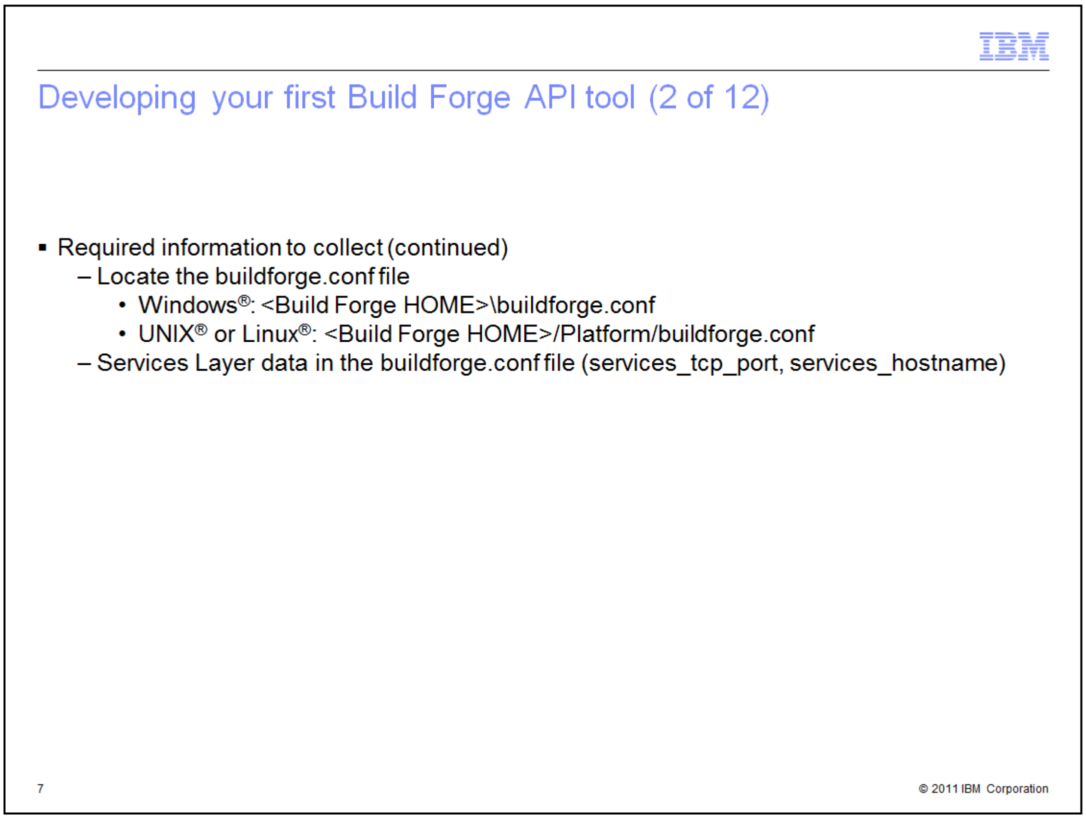
Developing your first Build Forge API tool (1 of 12)

- **What you need to get started:**
  - Java Runtime Environment V1.5 (Releases V7.1.1 and V7.1.2)
  - Java Runtime Environment V1.6 (Release V7.1.2 and Later)
  - Build Forge API Library (rbf-services-client-java.jar)
- **Required information to collect:**
  - Management Console information (User name and password)

© 2011 IBM Corporation

To get started with developing a Build Forge API tool, you will need a supported Java Runtime Environment (JRE). You will also need the Build Forge API library, which is in rbf-services-client-java.jar.

You will need to collect information required by the API tool for Build Forge Authentication and Services Layer Information. This includes the Management Console user name and password.



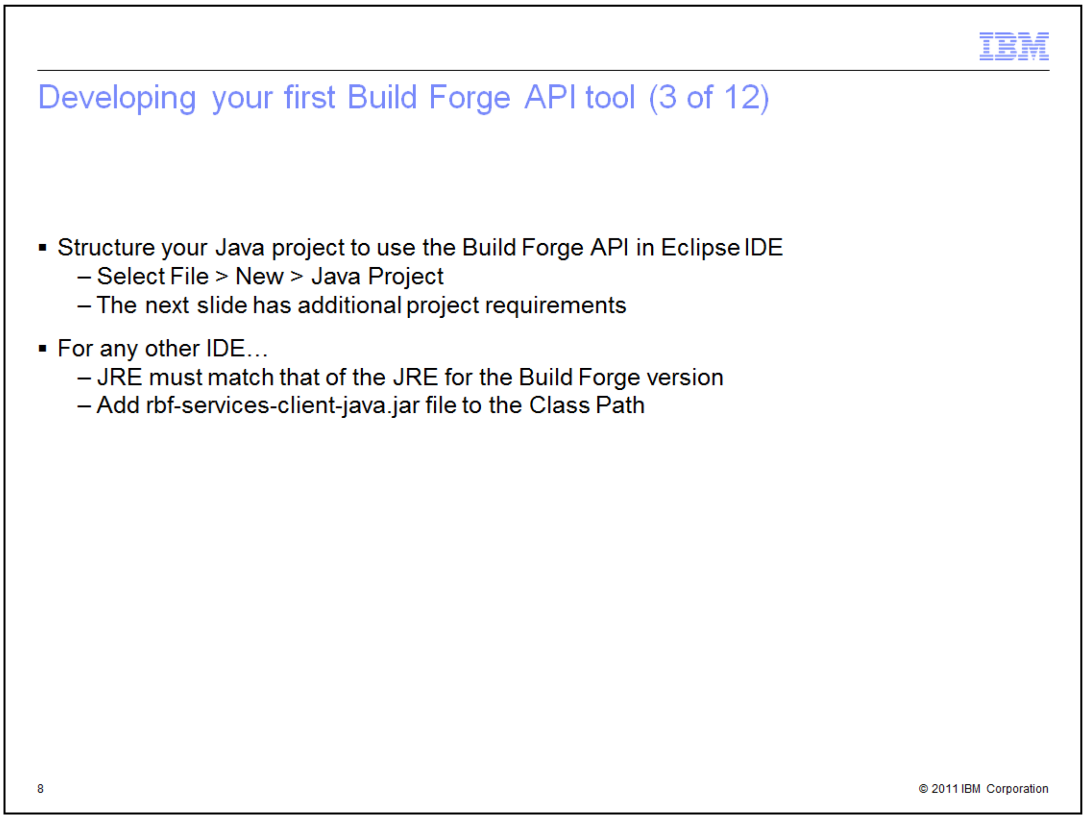
Developing your first Build Forge API tool (2 of 12)

- Required information to collect (continued)
  - Locate the buildforge.conf file
    - Windows®: <Build Forge HOME>\buildforge.conf
    - UNIX® or Linux®: <Build Forge HOME>/Platform/buildforge.conf
  - Services Layer data in the buildforge.conf file (services_tcp_port, services_hostname)

7

© 2011 IBM Corporation

You must also collect the Services Layer information that is located in the buildforge.conf file. In the file, you can locate the Services Layer Port value under services_tcp_port. You can find the Services Layer Hostname under services_hostname.



Developing your first Build Forge API tool (3 of 12)

- Structure your Java project to use the Build Forge API in Eclipse IDE
  - Select File > New > Java Project
  - The next slide has additional project requirements
- For any other IDE…
  - JRE must match that of the JRE for the Build Forge version
  - Add rbf-services-client-java.jar file to the Class Path

8 © 2011 IBM Corporation

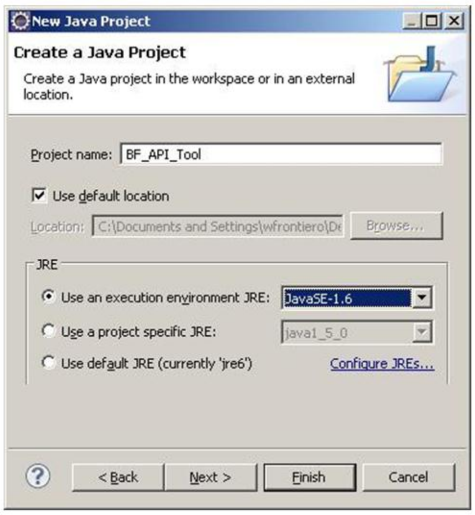
Now to structure your Java Project. In the Eclipse IDE, you create a new Java Project. You will create this project with a specific JRE and build path. Those values are explained in the next slide.

If you are using an alternative IDE, you must ensure that the JRE matches that of the JRE included in your version of Rational Build Forge. You must also add the rbf-services-client-java.jar file to the class path.

Developing your first Build Forge API tool (4 of 12)

- Select a Java Runtime Environment as the Eclipse Project JRE

Figure: (1.1)



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: BF_API_Tool

☑ Use default location

Location: C:\Documents and Settings\wfrontiero\De

JRE

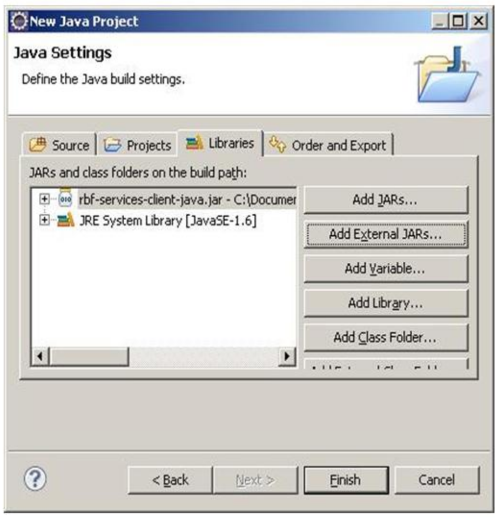
(●) Use an execution environment JRE: JavaSE-1.6

( ) Use a project specific JRE: java1_5_0

( ) Use default JRE (currently 'jre6')

- Add Build Forge API Library to build path of the Eclipse Project

Figure: (1.2)



New Java Project

Java Settings

Define the Java build settings.

Libraries

JARs and class folders on the build path:

rbf-services-client-java.jar - C:\Documer

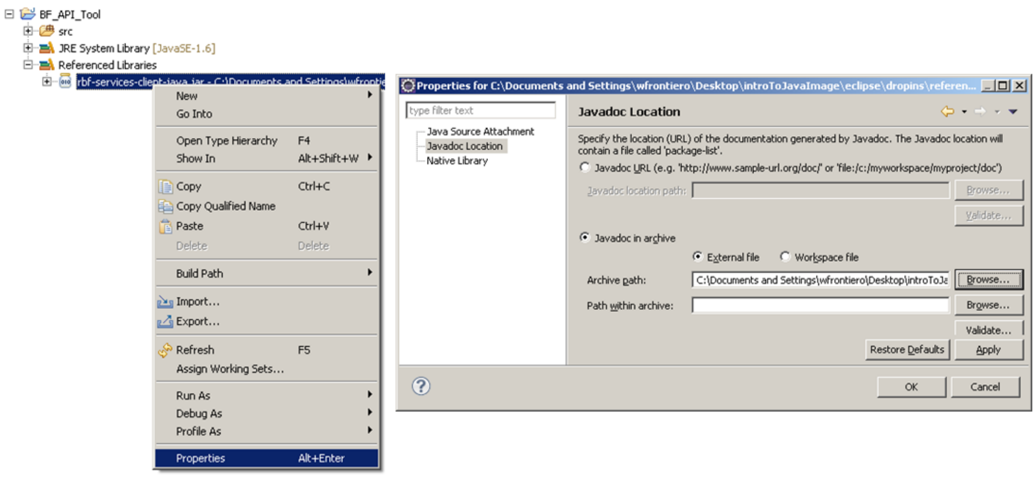
JRE System Library [JavaSE-1.6]

9 © 2011 IBM Corporation

To start developing your first Build Forge API tool, select a Java Runtime Environment that matches the JDK used by your Rational Build Forge server. To verify this, you can check the “ibmjdk” folder in the home directory on the server. After configuring a JRE for the project, you select the Libraries tab. Add the file rbf-services-client-java.jar to the build path.

Developing your first Build Forge API tool (5 of 12)

- Associate the rbf-services-client-java-doc.zip file to rbf-services-client-java.jar for JavaDoc access



Properties for C:\Documents and Settings\wfrontiero\Desktop\introToJavaImage\eclipse\dropins\referen...

Javadoc Location

Specify the location (URL) of the documentation generated by Javadoc. The Javadoc location will contain a file called 'package-list'.

○ Javadoc URL (e.g. 'http://www.sample-url.org/doc/' or 'file:/c:/myworkspace/myproject/doc')

Javadoc location path:

● Javadoc in archive

● External file ○ Workspace file

Archive path: C:\Documents and Settings\wfrontiero\Desktop\introToJa

Path within archive:

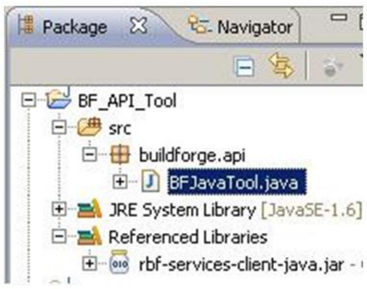
10

© 2011 IBM Corporation

Under your project, right click the Library. Click Properties. Select Javadoc Location on the navigation pane. Select the Javadoc in archive option. Browse to the rbf-services-client-java-doc.zip file. Click OK.

Developing your first Build Forge API tool (6 of 12)

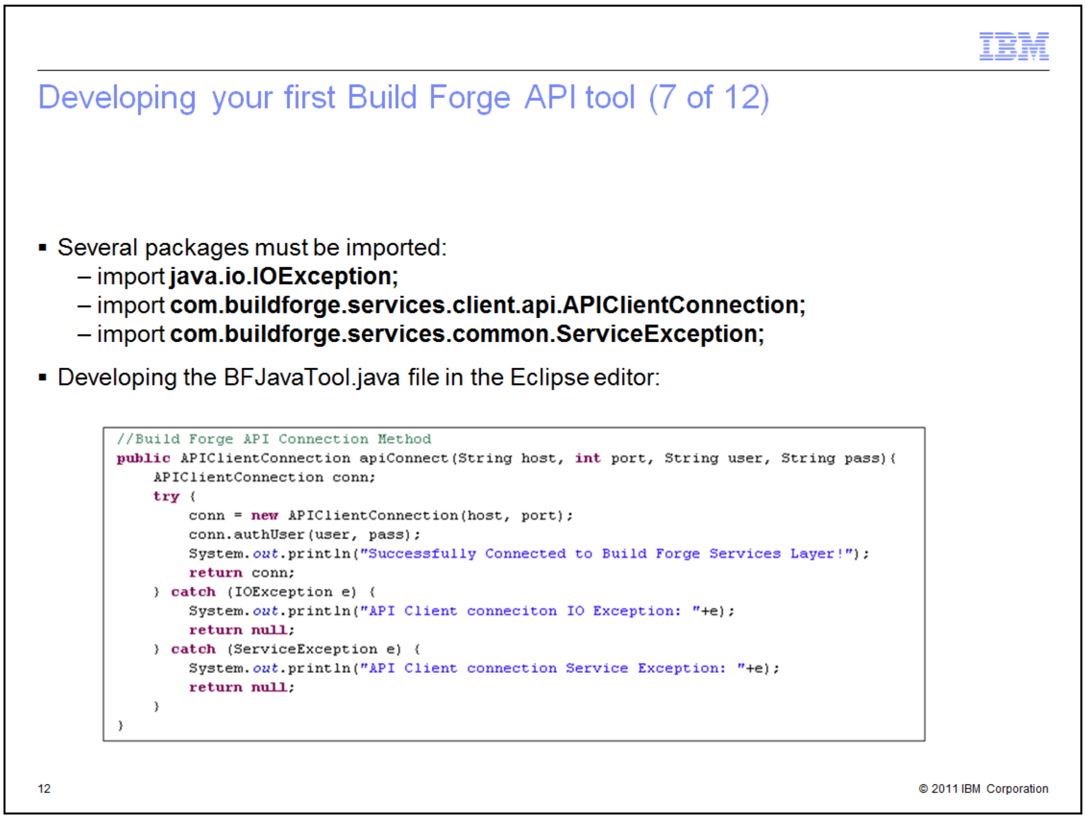
- Locate the newly Created Eclipse Project in the Package Navigator
- Here is what the newly created Java Project should now look like:



11 © 2011 IBM Corporation

For the remainder of this presentation, Eclipse Project structure is used to demonstrate he development of your first Build Forge API tool.

To begin work on your newly created Eclipse Project, open the Package Explorer within your Java Perspective. Right click the “src” folder. Create a new Package named “buildforge.api”. Right click the newly created “buildforge.api” package. Create a new Class. Name the Class BFJavaTool.



Developing your first Build Forge API tool (7 of 12)

- Several packages must be imported:
  - import **java.io.IOException;**
  - import **com.buildforge.services.client.api.APIClientConnection;**
  - import **com.buildforge.services.common.ServiceException;**
- Developing the BFJavaTool.java file in the Eclipse editor:

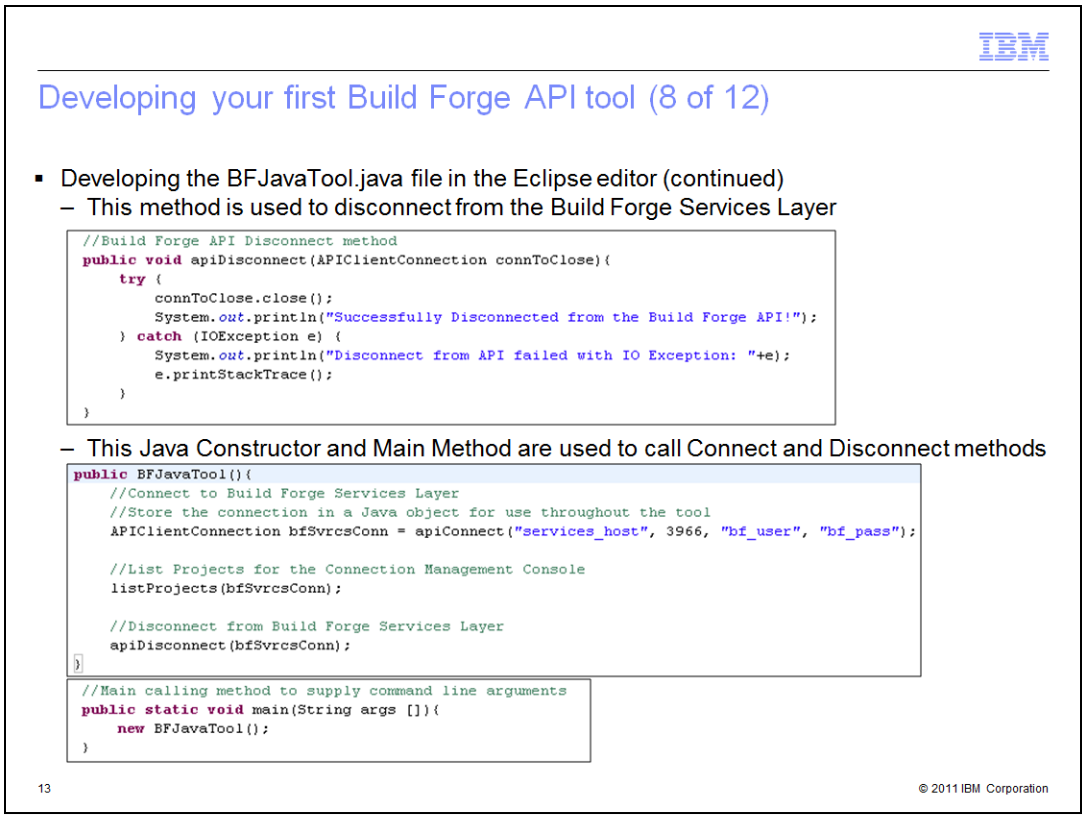
```
//Build Forge API Connection Method
public APIClientConnection apiConnect(String host, int port, String user, String pass){
    APIClientConnection conn;
    try {
        conn = new APIClientConnection(host, port);
        conn.authUser(user, pass);
        System.out.println("Successfully Connected to Build Forge Services Layer!");
        return conn;
    } catch (IOException e) {
        System.out.println("API Client conneciton IO Exception: "+e);
        return null;
    } catch (ServiceException e) {
        System.out.println("API Client connection Service Exception: "+e);
        return null;
    }
}
```

12

© 2011 IBM Corporation

These three packages must be imported in any Java class performing a Services Layer Connection. Add the code example seen here to make such Service Layer Connections.

As you can see in the example, “APIClientConnection” is a reusable Services Layer Connection object. You first pass the host and port to establish a TCP Connection. You then send the username and password authentication information to create the object.



Developing your first Build Forge API tool (8 of 12)

- Developing the BFJavaTool.java file in the Eclipse editor (continued)
  – This method is used to disconnect from the Build Forge Services Layer

```
//Build Forge API Disconnect method
public void apiDisconnect(APIClientConnection connToClose){
    try {
        connToClose.close();
        System.out.println("Successfully Disconnected from the Build Forge API!");
    } catch (IOException e) {
        System.out.println("Disconnect from API failed with IO Exception: "+e);
        e.printStackTrace();
    }
}
```

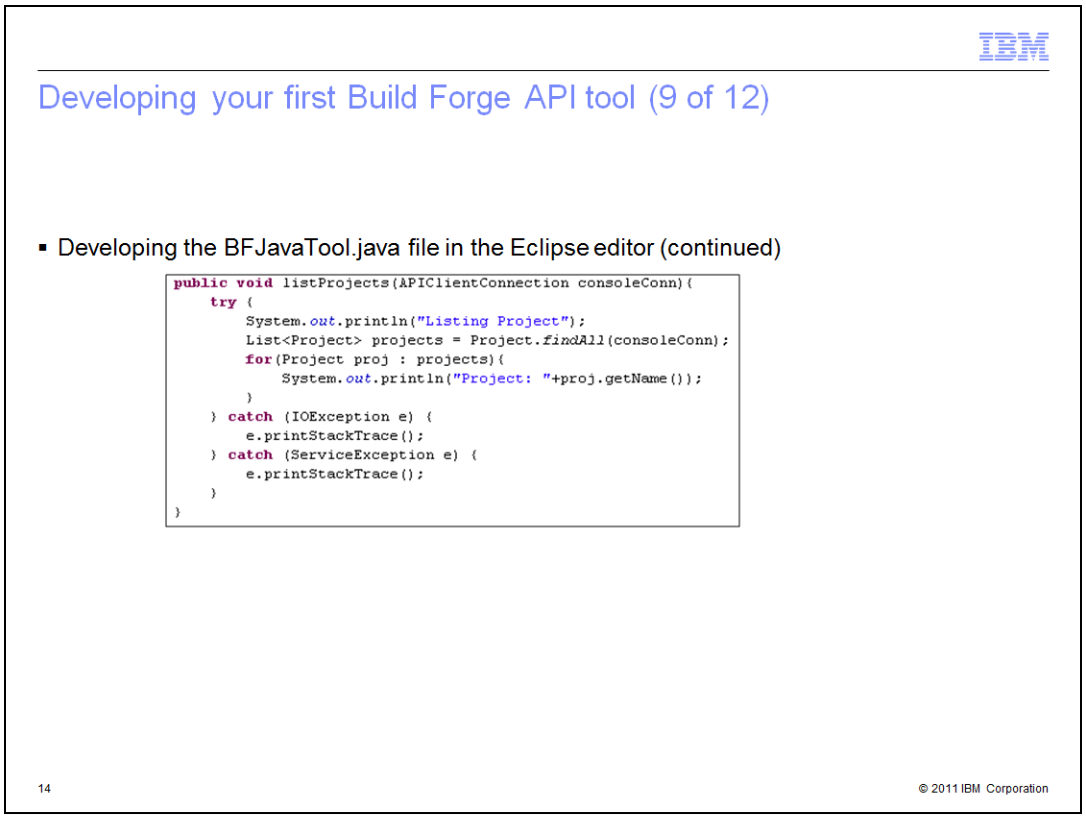
  – This Java Constructor and Main Method are used to call Connect and Disconnect methods

```
public BFJavaTool(){
    //Connect to Build Forge Services Layer
    //Store the connection in a Java object for use throughout the tool
    APIClientConnection bfSvrcsConn = apiConnect("services_host", 3966, "bf_user", "bf_pass");

    //List Projects for the Connection Management Console
    listProjects(bfSvrcsConn);

    //Disconnect from Build Forge Services Layer
    apiDisconnect(bfSvrcsConn);
}
```

```
//Main calling method to supply command line arguments
public static void main(String args []){
    new BFJavaTool();
}
```

13 © 2011 IBM Corporation

Here is how you complete the methods used to generate a complete APIClientConnection object. Also included is how to pass the APIClientConnection object to a method used to terminate the session.



Developing your first Build Forge API tool (9 of 12)

- Developing the BFJavaTool.java file in the Eclipse editor (continued)

```
public void listProjects(APIClientConnection consoleConn){
    try {
        System.out.println("Listing Project");
        List<Project> projects = Project.findAll(consoleConn);
        for(Project proj : projects){
            System.out.println("Project: "+proj.getName());
        }
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ServiceException e) {
        e.printStackTrace();
    }
}
```

14

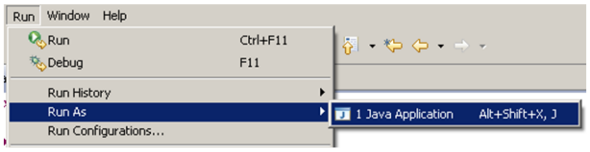
© 2011 IBM Corporation

Once a connection is established, you can perform any required task. In this example, you can list all projects in the Connected Build Forge Console. You start by importing necessary Packages for the Project object. Next, import com.buildforge.services.client.dbo.Project. You then import java.util.List for collecting the Project Object.

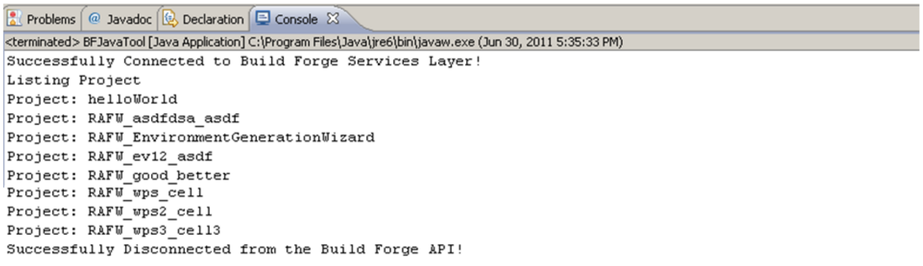
This Java code example uses the Services Connection to find all Project Objects returned as a Java List. After creating a List of Projects, you iterate through the List. With each iteration, you print each project name. Note that the Services Exception Catch block is required. It is necessary when making Service Layer calls from Java.

Developing your first Build Forge API tool (10 of 12)

- Developing the BFJavaTool.java file in the Eclipse editor (continued)
  - Run the application by selecting **Run as > Java Application** from **Run** menu



  - Console output from running BFJavaTool.java in Eclipse (Connects, Lists, Disconnects)

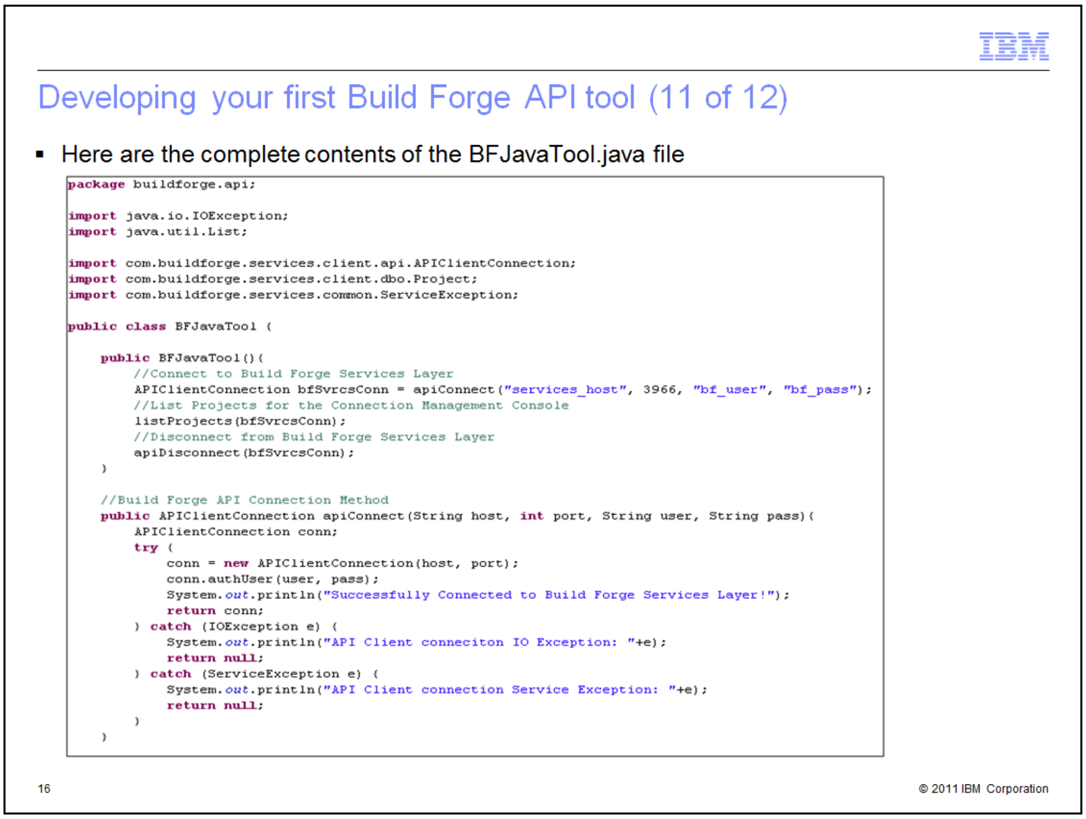


```
<terminated> BFJavaTool [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Jun 30, 2011 5:35:33 PM)
Successfully Connected to Build Forge Services Layer!
Listing Project
Project: helloWorld
Project: RAFW_asdfdsa_asdf
Project: RAFW_EnvironmentGenerationWizard
Project: RAFW_ev12_asdf
Project: RAFW_good_better
Project: RAFW_wps_cell
Project: RAFW_wps2_cell
Project: RAFW_wps3_cell3
Successfully Disconnected from the Build Forge API!
```

15

© 2011 IBM Corporation

Here is an example of running the file in the Eclipse editor and the output you will see in the console output.



Developing your first Build Forge API tool (11 of 12)

- Here are the complete contents of the BFJavaTool.java file

```java
package buildforge.api;

import java.io.IOException;
import java.util.List;

import com.buildforge.services.client.api.APIClientConnection;
import com.buildforge.services.client.dbo.Project;
import com.buildforge.services.common.ServiceException;

public class BFJavaTool {

    public BFJavaTool(){
        //Connect to Build Forge Services Layer
        APIClientConnection bfSvrcsConn = apiConnect("services_host", 3966, "bf_user", "bf_pass");
        //List Projects for the Connection Management Console
        listProjects(bfSvrcsConn);
        //Disconnect from Build Forge Services Layer
        apiDisconnect(bfSvrcsConn);
    }

    //Build Forge API Connection Method
    public APIClientConnection apiConnect(String host, int port, String user, String pass){
        APIClientConnection conn;
        try {
            conn = new APIClientConnection(host, port);
            conn.authUser(user, pass);
            System.out.println("Successfully Connected to Build Forge Services Layer!");
            return conn;
        } catch (IOException e) {
            System.out.println("API Client conneciton IO Exception: "+e);
            return null;
        } catch (ServiceException e) {
            System.out.println("API Client connection Service Exception: "+e);
            return null;
        }
    }
```

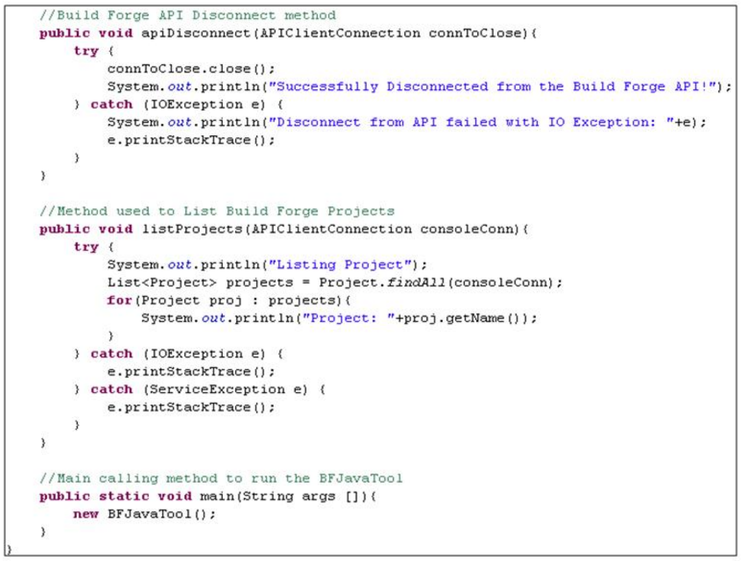
16

© 2011 IBM Corporation

In these next two slides, you will see the complete contents of the BFJavaTool.Java file.

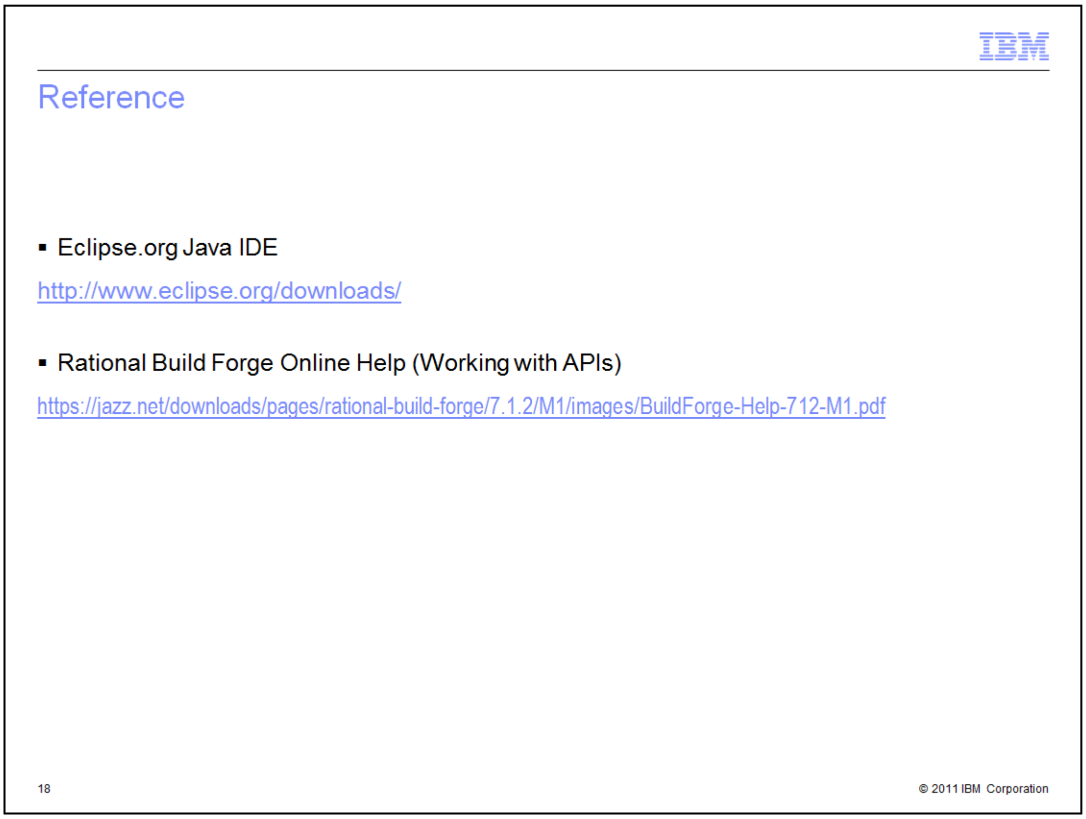
Developing your first Build Forge API tool (12 of 12)

- Here are the complete contents of the BFJavaTool.java file (continued)



```
    //Build Forge API Disconnect method
    public void apiDisconnect(APIClientConnection connToClose){
        try {
            connToClose.close();
            System.out.println("Successfully Disconnected from the Build Forge API!");
        } catch (IOException e) {
            System.out.println("Disconnect from API failed with IO Exception: "+e);
            e.printStackTrace();
        }
    }

    //Method used to List Build Forge Projects
    public void listProjects(APIClientConnection consoleConn){
        try {
            System.out.println("Listing Project");
            List<Project> projects = Project.findAll(consoleConn);
            for(Project proj : projects){
                System.out.println("Project: "+proj.getName());
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ServiceException e) {
            e.printStackTrace();
        }
    }

    //Main calling method to run the BFJavaTool
    public static void main(String args []){
        new BFJavaTool();
    }
}
```

© 2011 IBM Corporation



Reference

- Eclipse.org Java IDE

http://www.eclipse.org/downloads/

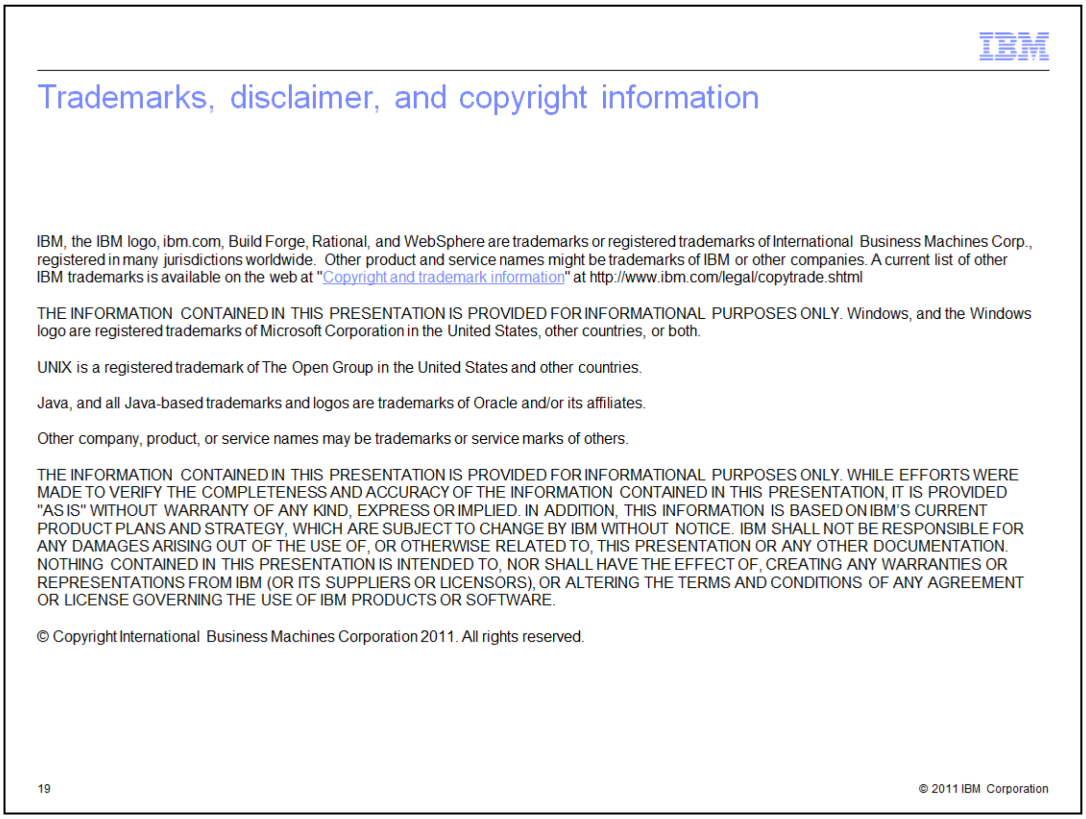
- Rational Build Forge Online Help (Working with APIs)

https://jazz.net/downloads/pages/rational-build-forge/7.1.2/M1/images/BuildForge-Help-712-M1.pdf

18

© 2011 IBM Corporation

Refer to these two links for information on Eclipse Java Integrated Development Environments and Rational Build Forge Online Help. The online help has a “Working with APIs” topic with more API information.



Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, Build Forge, Rational, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java, and all Java-based trademarks and logos are trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2011. All rights reserved.

© 2011 IBM Corporation