DEVCON 16

Creating real-world Building Blocks for your Institution: Part I

Session - WKS46371

Michael Bechtel / Jonathan Hart
Indian River State College

Blackboard

#BbWorld16

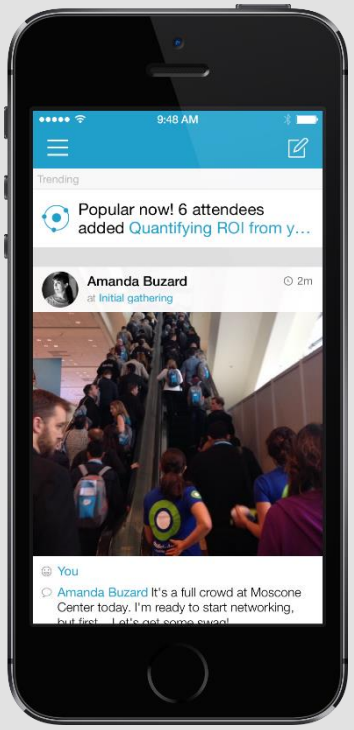
Download the BbWorld & DevCon 2016 Mobile App!

DEVCON 16

Available on the App Store

ANDROID APP ON Google play

HTML



Session - WKS46371

Be sure to **RATE** us!

Blackboard

#BbWorld16

Overview

- Environment Setup: Installing Oracle Virtual Box, Vagrant, and the Bb Learn Development VM
- Alt-Installing/Configuring Java on the Host Machine
- Choosing an IDE and configuring the JDK and Build Tools
- Downloading the bb-starting-block and installing it on the development VM for deployment
- Creating a new B2 project: Basic setup, settings and configurations, and layout.
- Deploying your B2 to the Development VM

Environment Setup

Environment Setup

- This walk through will be using Virtual Box: https://www.virtualbox.org/wiki/Downloads
- Install Vagrant: https://www.vagrantup.com/downloads.html
- From Behind the Blackboard: https://behind.blackboard.com/downloads/details.aspx?d=1669
  - Choose appropriate version: this presentation uses the Start Block 1.3 SP10 and Up
  - Download Development VM: October 2014: (Used in this presentation)
  - https://behind.blackboard.com/System-Administrator/Learn/Downloads/download.aspx?d=1714
- Hello World Git Repo for this presentation: https://github.com/IndianRiverStateCollege/irsc-hello-world
- **READ AND BOOKMARK THE FOLLOWING LINKS!!!:**
  - https://community.blackboard.com/community/developers/learn (Bb Community JOIN TODAY!!!)
  - https://community.blackboard.com/docs/DOC-1113 (Getting Started with Building Blocks)
  - https://community.blackboard.com/docs/DOC-1104 (Developer Virtual Machine)
  - https://community.blackboard.com/docs/DOC-1139 (First Project: Gradle, Git, & b2-starter-template)
  - https://community.blackboard.com/docs/DOC-1125 (Building Blocks and Java Permissions)
  - https://community.blackboard.com/docs/DOC-1115 **(Building Block API Documentation: API/TagLib/Schema)**
  - https://bbprepo.blackboard.com/content/repositories/releases/blackboard/platform/bb-schema-xsd/9.1.201410.160373/bb-schema-xsd-9.1.201410.160373.xsd (bb-schema for this presentation)
  - https://bbprepo.blackboard.com/content/repositories/releases/blackboard/platform/bb-manifest-plugin/9.1.201410.160373/bb-manifest-plugin-9.1.201410.160373.xsd (bb-manifest for this presentation)

Post Developer VM Installation

- Once the development environment is up and running, log into the VM

```
[vagrant@localhost ~]$ java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
[vagrant@localhost ~]$
```

- Depending on your Host Environment, proceed to download and install the Java JDK version that matches the VM.
  - This is so when you deploy your B2 and use Java Code, it will compile using the same version to avoid errors later on.
- Once you have the matching Java installed on your host environment you can now setup your IDE of choice.

IntelliJ by JetBrains

- https://www.jetbrains.com/idea/
- There are two versions: Ultimate (Paid) or Community Edition (Free)
- For the needs of Blackboard Development the CE version will do.
- The Ultimate Version does offer more tools and is easier than Eclipse, but lacks the efficiency that Eclipse offers when it comes to Maven.

Eclipse and the B2 Plugin

- https://www.eclipse.org/downloads/
- http://www.edugarage.com/display/BBDN/Building+Blocks+Eclipse+Plugin
- After choosing the appropriate Eclipse Application and the B2 plugin you can now start creating a B2

Spring Tool Suite (Eclipse)

- https://spring.io/tools/sts/all
- After installing STS and cloning the git repo, you will might have to change the web.xml

From:

```
<web-app version="2.5"
         xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
```

To:

```
<web-app version="2.5"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
```

Gradle Setup

- After following the documentation of downloading the Getting Started B2, you will need to change the “deployServer” to match the port that the VM is listening on: I.E: 9876

```
ext {
  learnVersion = "9.1.120113.0"
  deployServer = "localhost:9876"
  deployInstance = "BBLEARN"
}
```

If you do not know what this is, you can view the vagrant file in the VM's directory on the Host Machine

- 8080: http://localhost:9876
- 8443: SSL: https://localhost:9877
- 2222: SSH Tunnel: ssh vagrant@localhost -p2222
  - Note that you should use the vagrant ssh command instead, but if you need to have your IDE connection to do jobs or cleanup for certain things then you will may have to use this method.
- 5432: PostgreSQL, If your IDE has Database Connection support, then you can tie this into your project and view changes live, or use to gather intel on a Bb Item.
- Note that some applications do not connection to the forwarding port, you may have to use the actual listening port: 8080, 8443, 2222, or 5432

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  config.vm.provider :virtualbox do |vb|
    vb.customize ["modifyvm", :id, "--memory", "4098"]
  end

  config.vm.box = 'bb-learn-9.1.201410.160373'
  config.vm.box_url = './bb-learn-9.1.201410.160373.box'

  config.vm.network :forwarded_port, guest: 8080, host: 9876
  config.vm.network :forwarded_port, guest: 8443, host: 9877
  config.vm.network :forwarded_port, guest: 2222, host: 9878
  config.vm.network :forwarded_port, guest: 5432, host: 9879
end
```

Gradle Setup Continued

- Uncomment the API line, this will allow you include third party packages or your own and have them placed in the proper location

```
// Uncomment if your project has an API JAR
apply from: "api.gradle"
```

Additional Gradle documentation:
https://docs.gradle.org/current/userguide/userguide.html

Recommended Reading - Gradle Daemon:
https://docs.gradle.org/current/userguide/gradle_daemon.html

DEVCON 16

Creating A Blackboard B2

Creating a Blackboard B2

First we will go over the basic layout format of this particular B2: Hello World

After installing the basic-b2-template, we need to take a look at the starting layout and where things are going to be placed

```
.
├── api.gradle
├── build.gradle
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradlew
├── gradlew.bat
├── README.md
└── src
    ├── api
    │   └── java
    │       └── README.md
    ├── main
    │   ├── java
    │   │   └── README.md
    │   └── webapp
    │       ├── schema
    │       │   └── main
    │       │       └── schema.xml
    │       └── WEB-INF
    │           ├── bb-manifest.xml
    │           ├── bundles
    │           │   └── bb-manifest-en_US.properties
    │           └── web.xml
    └── test
        └── java
            └── README.md
```

*.gradle files

- The api.gradle file contains the configuration for external api (jar) or java code files. These settings will be injected during the: gradle deployB2 process

- The build.gradle file contains:
  - main build configuration
  - setup for the deployment server (VM)
  - naming of the war file
  - any additional compiling options

The build.gradle config

Main settings to focus on: **apply, group, version, and ext**

- **Apply**: apply from: “api.gradle” is not needed for the first deployment phase of the walk-through, but may be needed for some *.jar files and java code implemented to compile correctly. To enable/turn on: uncomment the line.

- **Group**: category of where your plugin will be placed under on the deployment server

- **Version**: version suffix of the B2, also applied to naming of the [plugin-name].war file: e.g: *[plugin-name]-[version].war*

- **Ext**:
  - LearnVersion: minimal Bb required version for this plugin
  - DeployServer: development server address
    - e.g: localhost:9876
  - DeployInstance: name of the App instance, which reflects the DB Schema name. *Unless configured differently, will be BBLEARN*

The api.gradle file

- Allows control of how external resources are packed into the file war file.
- Can include other path locations to include API jar files or sources here.

Folder Structures

The src folder structure

- **src**: main folder location gradle expects when compiling webapp
- **src/api**: houses external api jars or java code to be compiled and/or included into webapp at compilation
  Depending on build.gradle configurations, may or may not be included.
- **src/main**: main plugin code, resources, properties
- **src/test**: unit testing area for java code, jars, libs, extra.

Basic src/main folder structure

This area is very customizable to the developers preference, but there are some conventions that must be followed to get the basic B2 up and running on your Bb instance.

- **main/java**: java code - Controllers and Servlets
- **main/webapp**: main code - jsps, properties for the manifest, xml, etc

Basic src/main structure - continued

webapp: There are some mandatory elements needed that help the plugin installation and operation on a Bb Instance: WEB-INF folder, bb-manifest.xml file, and the web.xml file. *Optionally and suggested convention: use bundles and property files.*

The property files are for localization and can make your plugin internationally expandable.

Although documentation may show that adding plugin keys for localization “works” in the bb-manifest-*.properties, it does not always work.

When setting up the bb-manifest.xml, accessing the key values housed in the bb-manifest-[locale].properties.

In this example, a folder called resources has been created along with settings.properties. This is for separation between plugin setup and plugin localization in the code.

```
└── src
    ├── api
    │   └── java
    │       └── README.md
    ├── main
    │   ├── java
    │   │   └── README.md
    │   ├── resources
    │   │   └── settings.properties
    │   └── webapp
    │       ├── admin
    │       │   ├── configure.jsp
    │       │   ├── info.jsp
    │       │   └── remove.jsp
    │       ├── module
    │       │   └── index.jsp
    │       ├── schema
    │       │   └── main
    │       │       └── schema.xml
    │       └── WEB-INF
    │           ├── bb-manifest.xml
    │           ├── bundles
    │           │   └── bb-manifest-en_US.properties
    │           └── web.xml
    └── test
        └── java
            └── README.md
```

main/schema folder

- **main/schema**: used when interacting with the database with custom objects.

main/schema.xml layout:

```
<xs:element name="schema">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="comment" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="table" maxOccurs="unbounded"/>
            <xs:element ref="pre-update" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="stored-procedures" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="functions" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="triggers" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="packages" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="views" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="post-schema-update" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="post-update" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" use="required" type="xs:string"/>
        <xs:attribute name="comment" use="optional" type="xs:string"/>
        <xs:attribute name="license" use="optional" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

See:
https://bbprepo.blackboard.com/content/repositories/releases/blackboard/platform/bb-schema-xsd/9.1.201410.160373/bb-schema-xsd-9.1.201410.160373.xsd

MVC Structure with Java

MVC Structure with Java

- Blackboard is essentially a webapp that allows webapps to extend off its core platform.
- Each Bb webapp has ability to include java code and run as a separate webapp within Blackboard.
- Java code is placed under the src/main/java directory & is where you can create a full on MVC structured application.
- More on this in the Expanding with Java MVC section

Prior to Moving Forward

- Make sure to have the following installed:
  - IDE of Choice
  - Java JDK that matches the that of the Bb Dev VM
  - Maven
  - Git
  - Gradle
  - Depending on Setup: Ant

Creating the Hello World B2

Creating the Hello World B2

- Download the basic-b2-template from the git repo.
- The project name used in this example: `irsc-hello-world`
- Use git command from getting started page in the Bb Docs:
  ```
  git clone https://github.com/blackboard/basic-b2-template.git -o irsc-hello-world irsc-hello-world
  ```
- Git command breakdown: git clone <repo.url> -o <origin name> <repo.directory.name>

Making some necessary modifications

- Open the build.gradle and modify the following settings:
  - Uncomment the **apply** from: “api.gradle”
  - Change **group** to fit your institution/company.
  - Modify **version** to your specifications
  - Change the **ext** settings, e.g.:
    - **LearnVersion** = “9.1.120113”
    - **DeployServer** = “localhost:9876”
    - **DeployInstance** = “BBLEARN”

```
// Uncomment if your project has an API JAR
apply from: "api.gradle"

// define the version for the project when publishing to maven
group "edu.irsc"
version "1.0-BETA"

ext {
  learnVersion = "9.1.120113.0"
  deployServer = "localhost:9876"
  deployInstance = "BBLEARN"
}
```

Configuring WEB-INF/*

- Template provides the bundles and the properties file that is automatically included when compiled.
- Open bundles/bb-manifest-en_US.properties file and make the appropriate modifications.
- Names are self-explanatory

```
b2.name=Hello World
b2.description=Simple B2 example with Servlets
b2.vendor.description=Some vendor description stuff here

app1.handle=hello-world-app
app1.name=Hello World App
app1.description=Application Description
tool1.name=Hello World Tool
tool1.description=Hello World Example Description
content_handler.name=Hello World Custom Content Handler
```

Understanding the bb-manifest.xml

The bb-manifest.xml is the main configuration of the plugin or “webapp” if you will. This file does have a strict rule set in what is considered an known allowed node element. Below is just the minimal nodes that are needed for a simple B2:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns="http://www.blackboard.com/bb-manifest-plugin"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.blackboard.com/bb-manifest-plugin
https://maven.blackboard.com/content/repositories/releases/blackboard/platform/bb-manifest-plugin/9.1.110082.0/bb-manifest-plugin-
9.1.110082.0.xsd">
    <plugin>
        <name value="b2.name"/>
        <handle value="hello-world"/>
        <description value="b2.description"/>
        <default-locale value="en_US"/>
        <version value="1.0"/>
        <requires>
            <bbversion value="9.1.120123.0"/>
        </requires>
        <vendor>
            <id value="irsc"/>
            <name value="Indian River State College"/>
            <url value="http://www.irsc.edu"/>
            <description value="b2.vendor.description"/>
        </vendor>
        <!-- Code permissions required for proper operation -->
        <permissions>
            <permission type="attribute" name="user.authinfo" actions="get"/>
            <permission type="attribute" name="user.personalinfo" actions="get"/>
            <permission type="java.lang.RuntimePermission" name="db.connection.*" actions="connect,accept"/>
            <permission type="socket" name="*.irsc.edu" actions="connect,resolve"/>
            <permission type="socket" name="localhost" actions="connect,resolve"/>
            <!-- Below Seems to work well for Development, but should do:
                  <permission type="java.io.FilePermission" name="./logs/my_b2.log" actions="read,write"/>
                   once you have everything set in place. -->
            <permission type="java.io.FilePermission" name="&amp;lt;&amp;lt;ALL FILES&amp;gt;&amp;gt;"
actions="read,write,delete,execute"/>
        </permissions>
    </plugin>
</manifest>
```

How the properties file and bb-manifest.xml relate

- Only certain values are injected from the properties file into the main xml file. The main layout structure should at least follow this format:
  - Manifest
    - Plugin
      - Name
      - Handle
      - Description
      - Default-locale
      - Version
      - Requires
        - Bbversion
      - Vendor
        - Id
        - Name
        - Url
        - Description
      - Permissions
        - Permission types
- Copy and Paste the provided template on the next slide

bb-manifest.xml example

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns="http://www.blackboard.com/bb-manifest-plugin"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.blackboard.com/bb-manifest-plugin
https://maven.blackboard.com/content/repositories/releases/blackboard/platform/bb-manifest-plugin/9.1.110082.0/bb-manifest-
plugin-9.1.110082.0.xsd">
    <plugin>
        <name value="b2.name"/> <!-- Will come from the properties file -->
        <handle value="<HANDLE VALUE HERE>"/> <!-- Will NOT come from the properties file; must be filled in here -->
        <description value="b2.description"/> <!-- Will come from the properties file -->
        <default-locale value="en_US"/> <!-- Will NOT come from the properties file; must be filled in here -->
        <version value="<B2 VERSION HERE>"/> <!-- Will NOT come from the properties file; must be filled in here -->
        <requires>
            <bbversion value="<MINIMAL BB REQUIRED VERSION HERE>"/> <!-- Will NOT come from the properties file; must be filled in here -->
        </requires>
        <vendor>
            <id value="<4 CHAR ID>"/> <!-- Will NOT come from the properties file; must be filled in here -->
            <name value="<VENDOR NAME VALUE HERE>"/> <!-- Will NOT come from the properties file; must be filled in here -->
            <url value="<VENDOR URL HERE>"/> <!-- Will NOT come from the properties file; must be filled in here -->
            <description value="b2.vendor.description"/> <!-- Will come from the properties file -->
        </vendor>
        <!-- Code permissions required for proper operation -->
        <permissions>
            <permission type="attribute" name="user.authinfo" actions="get"/>
            <permission type="attribute" name="user.personalinfo" actions="get"/>
            <permission type="java.lang.RuntimePermission" name="db.connection.*" actions="connect,accept"/>
            <permission type="socket" name="*.<YOUR-DOMAIN>.<com|edu|what-not>" actions="connect,resolve"/>
            <permission type="socket" name="localhost" actions="connect,resolve"/>
            <!-- Below Seems to work well for Development, but should do:
                  <permission type="java.io.FilePermission" name="./logs/my_b2.log" actions="read,write"/>
                   once you have everything set in place. -->
            <permission type="java.io.FilePermission" name="&amp;lt;&amp;lt;ALL FILES&amp;gt;&amp;gt;"
actions="read,write,delete,execute"/>
        </permissions>
    </plugin>
</manifest>
```

The web.xml

- This is the “help” and meta-data for the webapp, and used to help provide information or direct the workflow of the application.

- We will add some servlet nodes later and use a servlet-mapping node to help direct a request to a controller and provide data for our view.

Building/Deploying B2 to VM

Building/Deploying B2 to VM

For a quick test, we will run the following command -
(From the project folder in the terminal or project emulated terminal; or even a run command provided by your IDE of choice):

gradle deployB2

You should see the following result. If not, please read the error stack trace and locate/correct the incorrect portion that is causing the error.

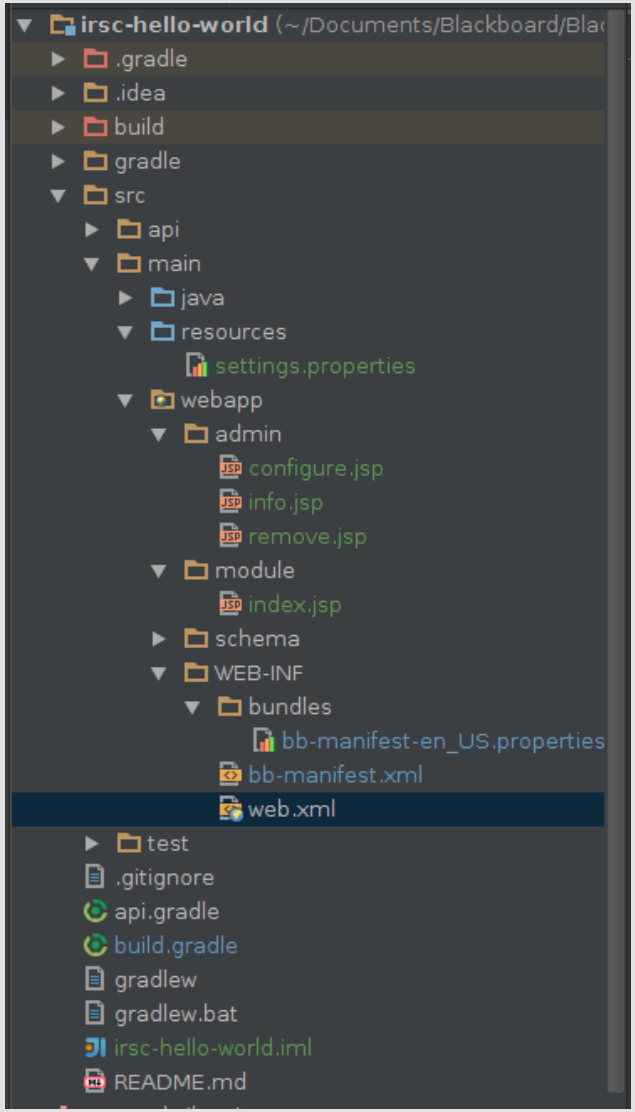
```
Starting a new Gradle Daemon for this build (subsequent builds will be faster).
:compileApiJava UP-TO-DATE
:processApiResources UP-TO-DATE
:apiClasses UP-TO-DATE
:apiJar
:compileJava UP-TO-DATE
:processResources
:classes
:war
:deployB2

BUILD SUCCESSFUL

Total time: 7.546 secs
```

This should build the project and create to folders:

- .gradle
- build



Building/Deploying B2 to VM

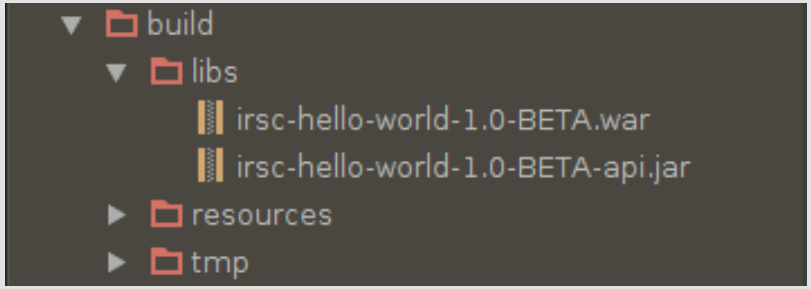
The build folder is created and contains a libs folder and the war and jar files for this B2.

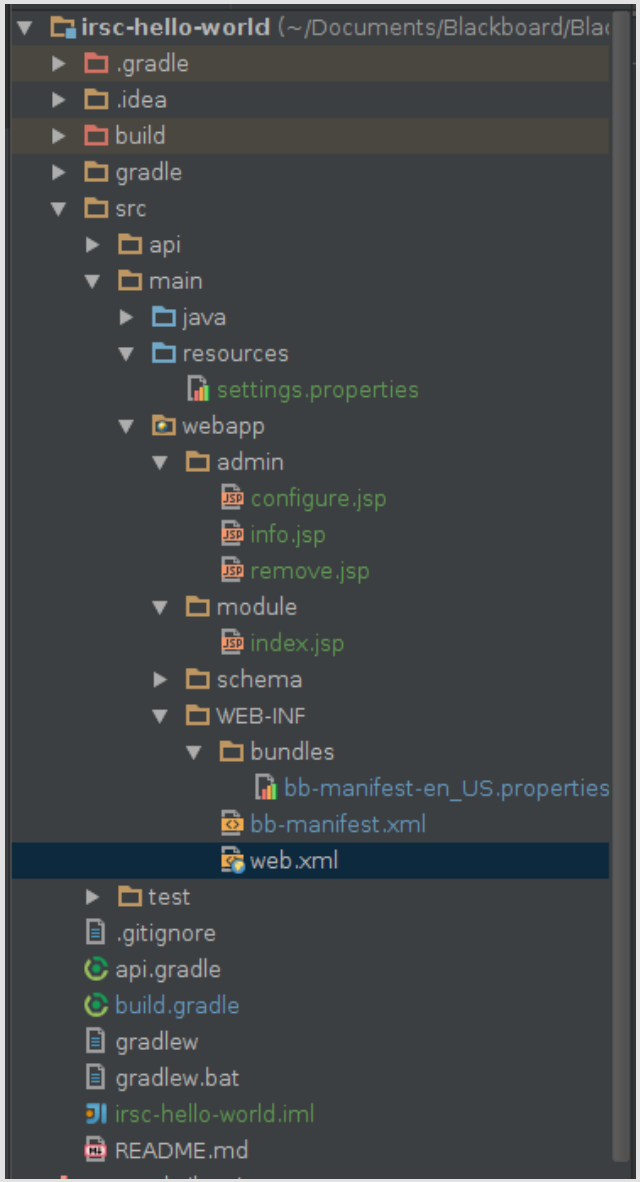
Please Note: If there is no jar file in the libs folder, this means the **apply** api line in the build.gradle file is not uncommented as explained earlier

If the apply api line is uncommented and there is no jar file, then something went wrong with the build. Trace down any errors and correct them.

The output is based off of the project name and the version setting from the build.gradle file.

The image below may look a little different, but will be covered in the next section.





Building/Deploying B2 to VM

If the deployment was successful:

- Open http://localhost:9876 in web browser
- Login
- System Admin
- Under the Building Blocks Panel >> Building Blocks
- Installed Tools
- Listed Alphabetically >> proceed to the Second Page or Show All and look for the Hello World Building Block

	Group Manager	Blackboard Inc.	1.4.1135109	Available
■	Hello World	Indian River State College	1.0	Available
■	HETemplateModules	Blackboard Inc.	9.2.1109065	Available

Back to the Code!

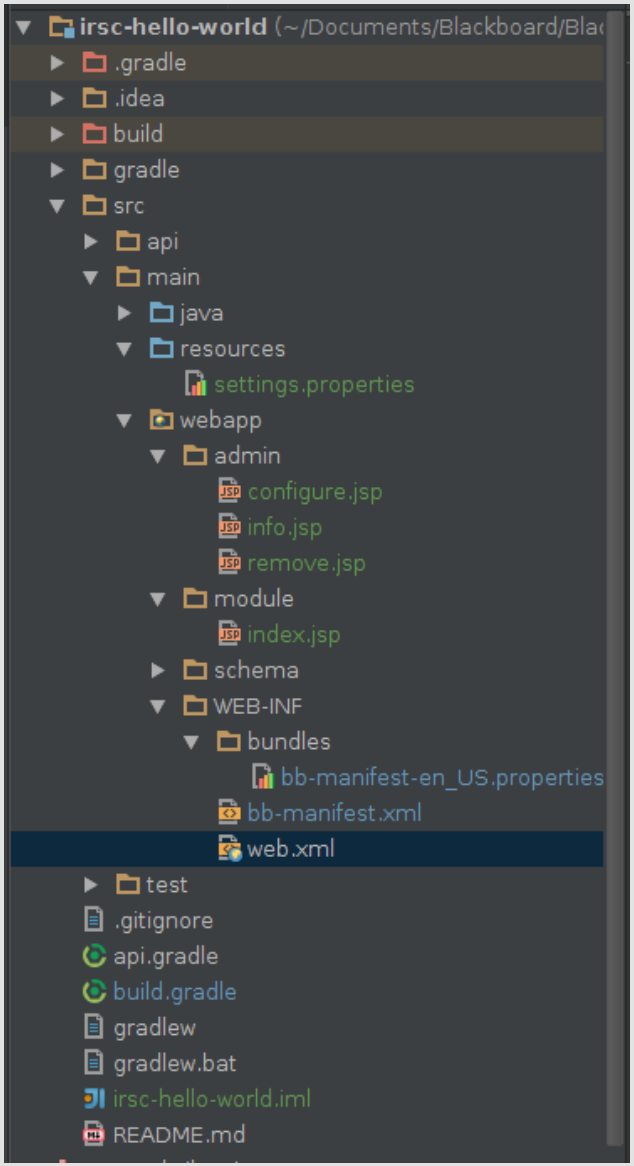
Back to the Code

On the previous pages, the layout was a little different.

- Created src/main/resources folder:
  - settings.properties file
- Created src/main/webapp/admin folder:
  - configure.jsp
  - info.jsp
  - remove.jsp
- Created src/webapp/module folder:
  - index.jsp

For now remove.jsp is an empty template page, but can be used to clean-up prior to the plugin being uninstalled from the server. This is meant to be a pre-post script that is ran just before the final deletion of the app on the server and it should not contain any UI elements and should be a non-interactive web page.

info.jsp is blank, but can be utilized later for in-house information on the app or documentation related to the B2 plugin.



Setting up index.jsp & adjusting the bb-manifest

- Create a folder in the src/webapp.
  - Please Note: this folder can be named anything desired. This example will use “module”.
- Create a file named “index.jsp” & paste the following code into the file.

```
<%@ taglib uri="/bbData" prefix="bbData"%>
<%@ taglib uri="/bbNG" prefix="bbNG"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<bbData:context>
    <bbNG:genericPage>

        <fmt:setBundle basename="settings" />
        <fmt:message key="name" var="plugin_name"/>
        <fmt:message key="hw.main.page.breadcrumb" var="plugin_breadcrumb"/>
        <fmt:message key="hw.main.page.header" var="page_title"/>
        <fmt:message key="hw.main.tabbed.panel" var="tabbed_panel"/>

        <bbNG:breadcrumbBar environment="SYS_ADMIN" navItem="admin_plugin_manage">
            <bbNG:breadcrumb href="${recallUrl}" title="${plugin_name}" />
            <bbNG:breadcrumb title="${plugin_breadcrumb}" />
        </bbNG:breadcrumbBar>

        <bbNG:pageHeader>
            <bbNG:pageTitleBar title="${page_title}">
                ${plugin_title}
            </bbNG:pageTitleBar>
        </bbNG:pageHeader>

        <bbNG:tabbedPanels>
            <bbNG:tabbedPanel title="${tabbed_panel}">
                ${tabbed_panel}
            </bbNG:tabbedPanel>
        </bbNG:tabbedPanels>

    </bbNG:genericPage>
</bbData:context>
```

The code breakdown: index.jsp

- Implement the tab libraries that Bb offers to utilize the look and feel of Bb Learn and inject custom code in to rendering hooks that are available.
  - **BbData**: Provides access to the DB elements & the current webapp context variables
  - **bbNG**: UI look & feel library that injects the proper markup at rendering
  - **jstl/core**: Core java syntax library providing programmatic logic to the webpage.
  - **jstl/fmt**: Formatting template library provided for loading data from settings bundle (properties file) for storing it into local variable(s) for use on the page allowing for dynamic/localized content.

Understanding the layout: index.jsp

- Create the **bbData:context** element which provides access to the current context.

- Create the **bbNG:genericPage** element which hooks the page into the main bb learn webapp and prevents it loading outside of main themed layout.
  (View the tablibs url provided in overview to learn more on page rendering hook elements)

- Create a **fmt:setBundle** element which loads the properties file created (please do so if you have not already) in the src/main/resources folder. Use the basename attribute and set that to the name of the properties file: "settings" in this example.

- Create several variables be creating **fmt:message** elements. The format for this is: <fmt:message key="*<key from prop>*" var="*<the var name to be created>*"/>. Follow Java conventional variable names: *(No spaces, must not start with a number or operational char, no dots, etc.)*

- To access these variables, use the basic template injection method: **`${variable}`**

- Please notice: **`${recallUrl}`** in this script. This is a global context variable that recalls the last page the user was on before loading this page.

The settings.properties file

The settings.properties file will be the main locale for this B2. It can be modified it to allow for localization by changing the name to:

bb-manifest-<locale>.properties

and then placing it in the bundles directory.

Property File Explanation

The left hand side is the “key” and the right hand side is the “value”. This is assigned by using the =(equals) symbol/operator in the file.

Use the # (hash|pound) symbol to denote comments

(Optional)

Common practice uses namespace style for keys

```
name=Hello World
title=Hello World Settings
header.instructions=Page Header Instructions



# Settings Page
step1.title=Step One
step1.label=Label for Data Element
step1.instructions=These are the instructions for: Step One
step1.helptext=This the Help Text for: Step One.


# Plugin Main Page
hw.main.page.header=Hello World Plugin Page Header
hw.main.page.breadcrumb=Hello World Page Breadcrumb
hw.main.tabbed.panel=Tabbed Panel Content
```

Setting up the B2 File Structure

Setting up the B2 File Structure

- Create **src/main/webapp/admin** folder and:
  - configure.jsp
  - remove.jsp
  - info.jsp – (optional) may not be used

Optional, but Recommended

- remove.jsp will be blank, serves as reminder when using custom database objects to clean them up.

Basic jsp Layout with variables

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ taglib uri="/bbNG" prefix="bbNG"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<bbNG:genericPage>
    <fmt:setBundle basename="settings" />
    <fmt:message key="name" var="plugin_name"/>
    <fmt:message key="title" var="page_title" />
    <fmt:message key="header.instructions" var="header_instructions" />
    <fmt:message key="step1.title" var="step1_title" />
    <fmt:message key="step1.label" var="step1_label" />
    <fmt:message key="step1.instructions" var="step1_instructions" />
    <fmt:message key="step1.helptext" var="step1_helptext" />
    <bbNG:breadcrumbBar environment="SYS_ADMIN" navItem="admin_plugin_manage">
        <bbNG:breadcrumb href="${recallUrl}" title="${plugin_name}" />
        <bbNG:breadcrumb title="${page_title}" />
    </bbNG:breadcrumbBar>
    <bbNG:pageHeader instructions="${header_instructions}">
        <bbNG:pageTitleBar showTitleBar="true" title="${page_title}" />
    </bbNG:pageHeader>
    <bbNG:form action="" id="id_simpleForm" name="simpleForm" method="post" onsubmit="return validateForm();">
        <bbNG:dataCollection markUnsavedChanges="true" showSubmitButtons="true">
            <bbNG:step hideNumber="true" id="stepOne" title="${step1_title}" instructions="${step1_instructions}">
                <bbNG:dataElement isRequired="true" label="${step1_label}">
                    <bbNG:textElement id="id" name="Name" value="Value" helpText="${step1_helptext}" size="50" minLength="1" />
                </bbNG:dataElement>
            </bbNG:step>
            <bbNG:stepSubmit hideNumber="true" showCancelButton="true" cancelUrl="${recallUrl}" />
        </bbNG:dataCollection>
    </bbNG:form>
</bbNG:genericPage>
```

Code breakdown: configure.jsp

- Optional - Provide the doctype for the page
- Typical taglib inclusions
  - bbNG:genericPage
- Setup and import keys from settings.properties file and insert them into variables
- Create basic layout and inject variables onto page.
- Remember: refer back to the taglib documentation:

https://community.blackboard.com/docs/DOC-1115

This Hello World B2 uses the 9.1 October 2014 Tag Library

Setup remove.jsp and info.jsp

- For now, use this code as a placeholder.

  If it was decided to create these files

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html><head><title>Title</title></head><body></body></html>
```

Re-configuring the bb-manifest.xml

- Add the following code snippet under the vendor node: (Note: if not using remove.jsp, do not include the remove node element)

```
<http-actions>
  <config value="admin/configure.jsp"/>
  <remove value="admin/remove.jsp"/>
</http-actions>
```

- Add the following code snippet directly under the </http-actions> element

```
<application-defs>
    <application
            handle="<replace-with-your-handle>"
            type="system"
            use-ssl="false"
            name="<Friendly name of the App>"
            can-allow-guest="false"
            small-icon=""
            large-icon="">
        <description lang="en_US">Application Description (Application.Description)</description>
        <links>
            <link>
                <type value="<tool_type>"/>
                <name value="<Friendly Tool App Name>"/>
                <url value="<entry point for the plugin>"/>
                <description value="app1.description"/> <!-- loaded from the bb-manifest-*.properties -->
            </link>
        </links>
    </application>
</application-defs>
```

Example <application-defs> in Hello World

```
<application-defs>
  <application
       handle="irsc-hello-app"
       type="system"
       use-ssl="false"
       name="Hello World App"
       can-allow-guest="false"
       small-icon=""
       large-icon="">
    <description lang="en_US">Application Description (Application.Description)
</description>
    <links>
      <link>
        <type value="system_tool"/>
        <name value="Hello World Tool"/>
        <url value="module/index.jsp"/>
        <description value="app1.description"/>
      </link>
    </links>
  </application>
</application-defs>
```

Add code under the application-defs element

```
 <module-defs>
  <module-type ext-ref="irschelloworld" title="Hello World Module" uicreatable="false">
   <jsp-dir>module</jsp-dir>
   <jsp>
         <admin>admin/admin.jsp</admin>
         <view>view.jsp</view>
   </jsp>
   </module-type>
   <module type="irschelloworld" isadmin="true" useraddable="true" isdeletable="true"
isdetachable="true" title="Hello World Module">
   <description>Hello World Module Description.</description>
   <ExtraInfo/>
   <module-groups>
    <module-group id="Other" />
   </module-groups>
  </module>
 </module-defs>
```

Note: change the content to fit the application: module-type ext-ref, title, jsp/admin and view values, etc. Use the above code snippet as a guide.

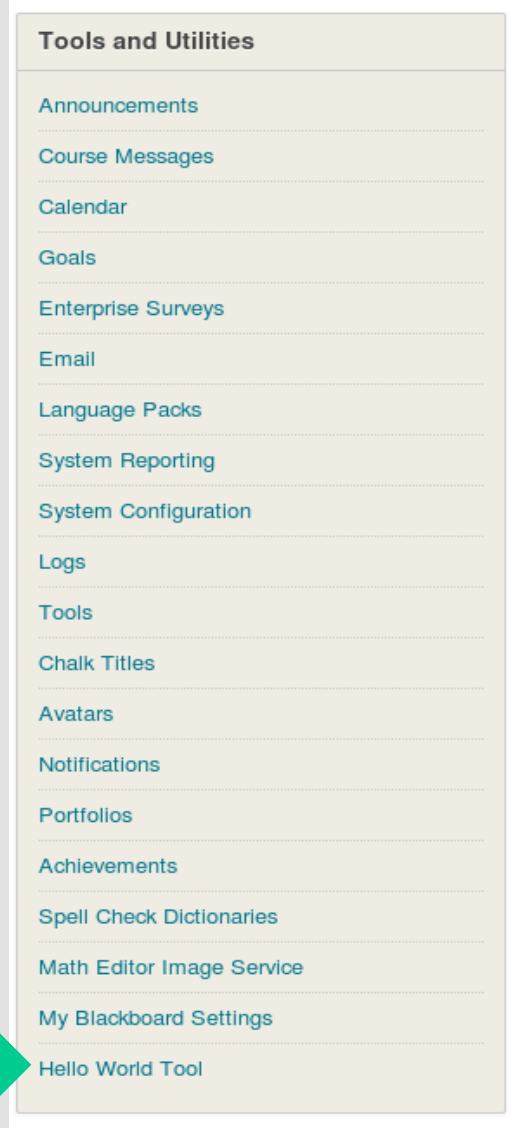
Rebuild using Gradle

- After changes are made, rebuild project in the terminal or whatever method used to build the project in the beginning:
  - e.g: "**`gradle deployB2`**"

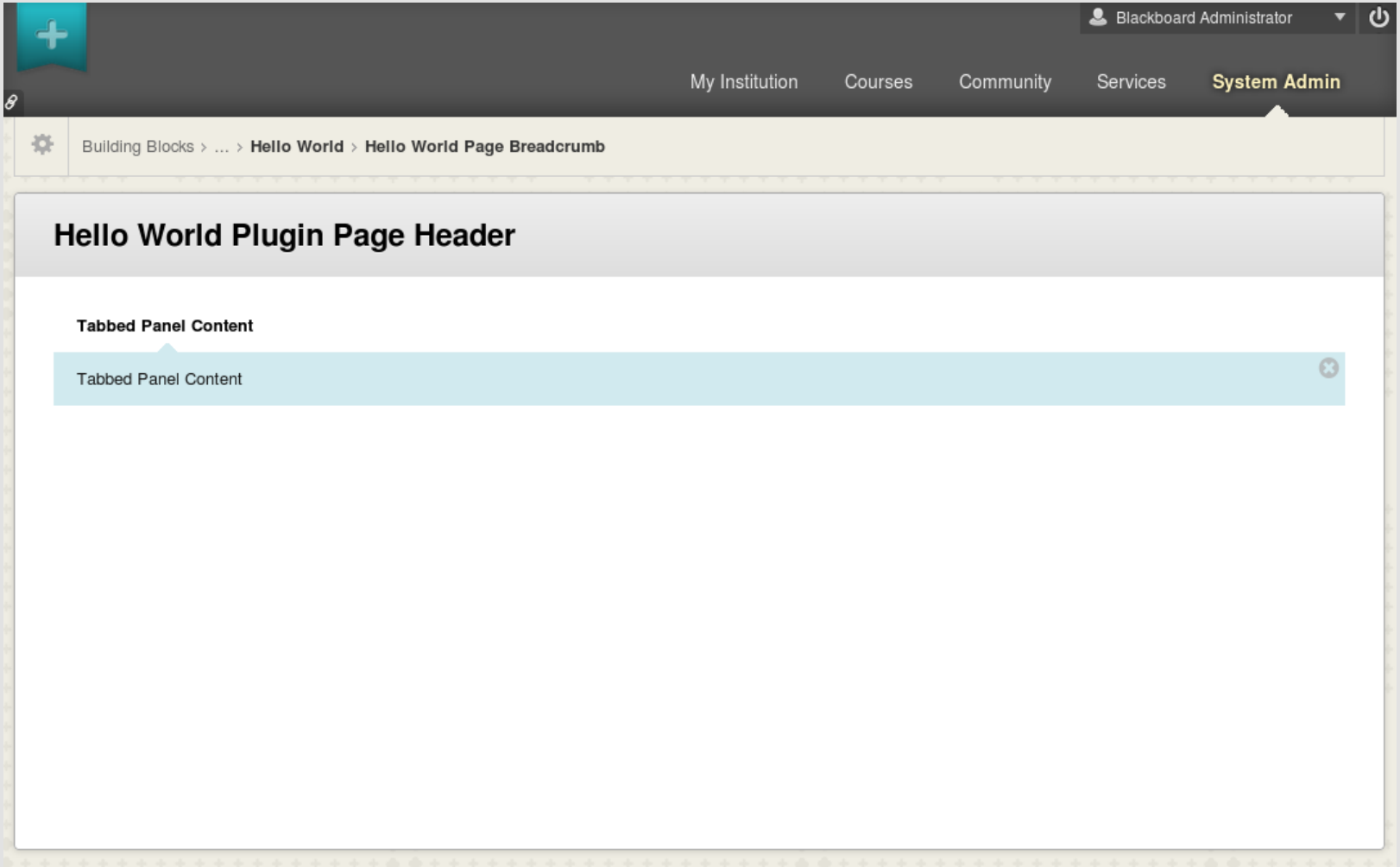
Results

- The tool is now located in the System Admin Panel under the Tools and Utilities listed as: Hello World Tool

- Change the name anytime in the bb-manifest.xml and after rebuild/deploy it will update here.

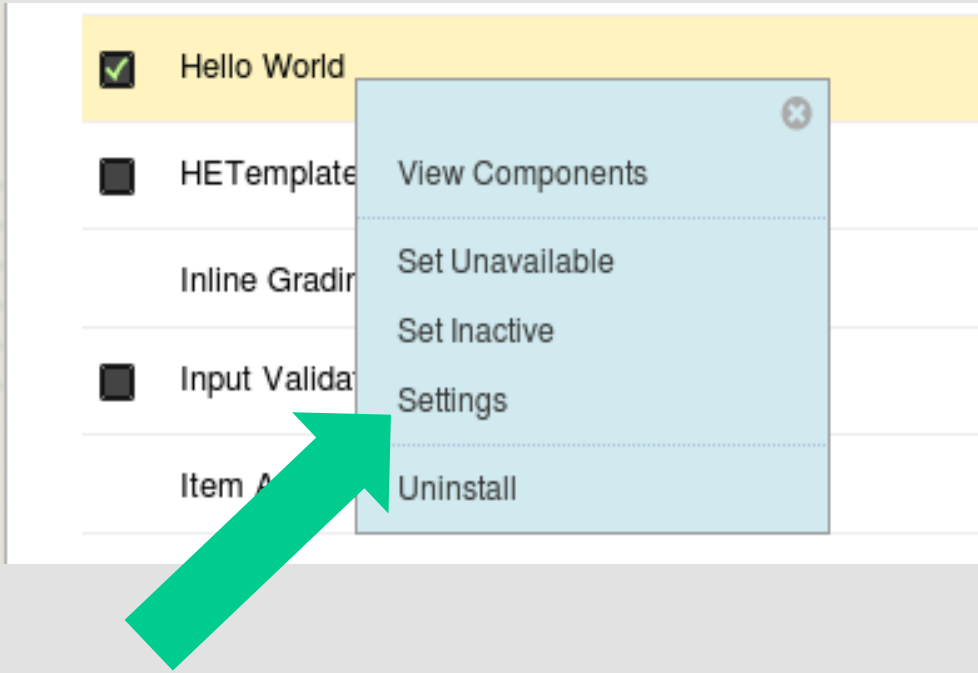


Hello World index.jsp in BbUI

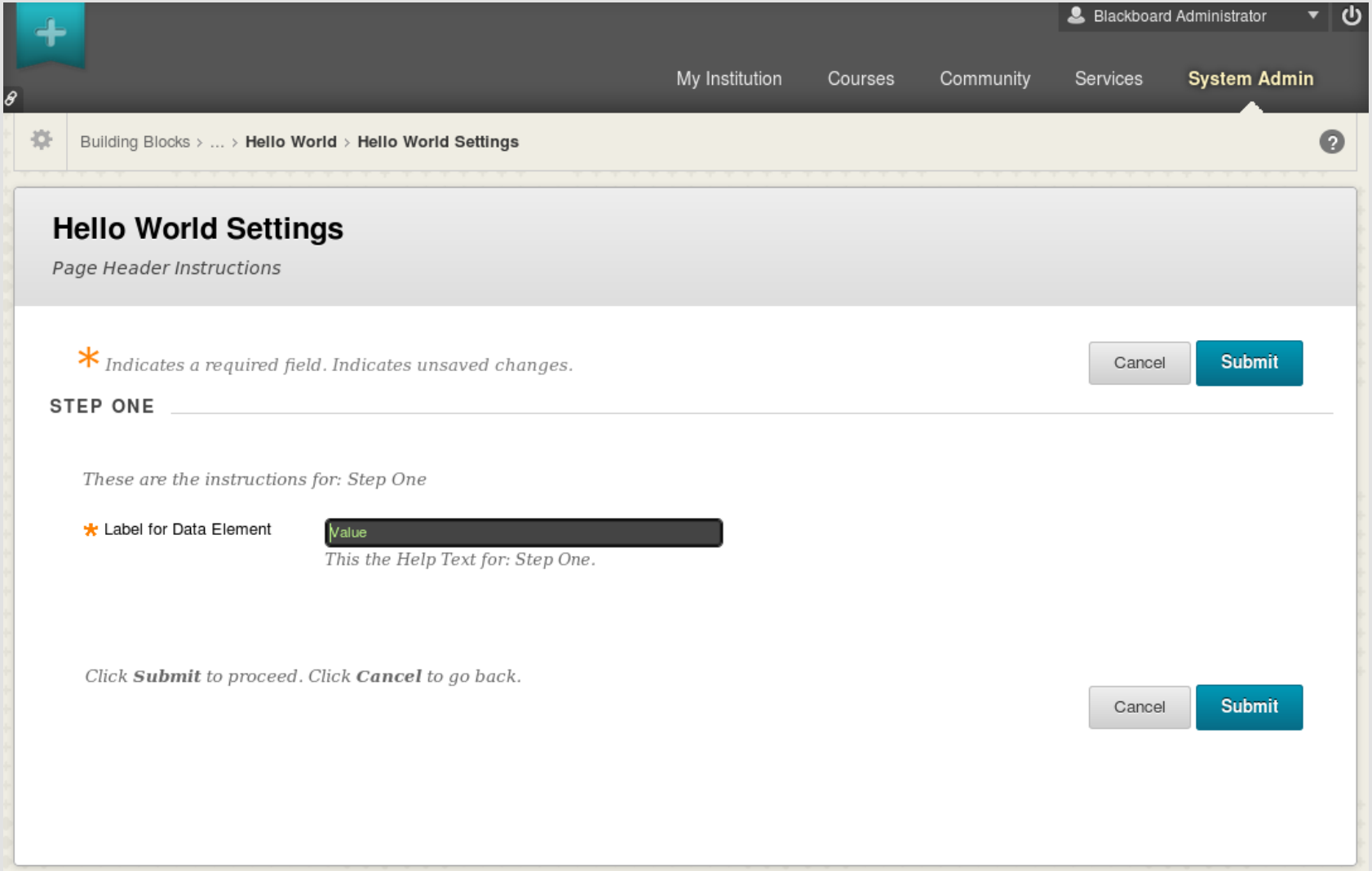


Settings Now Available!

- Go to
  - System Admin Panel
  - Building Blocks
  - Installed Building Blocks
  - Locate the B2
  - Click on the Chevron
  - Click on Settings



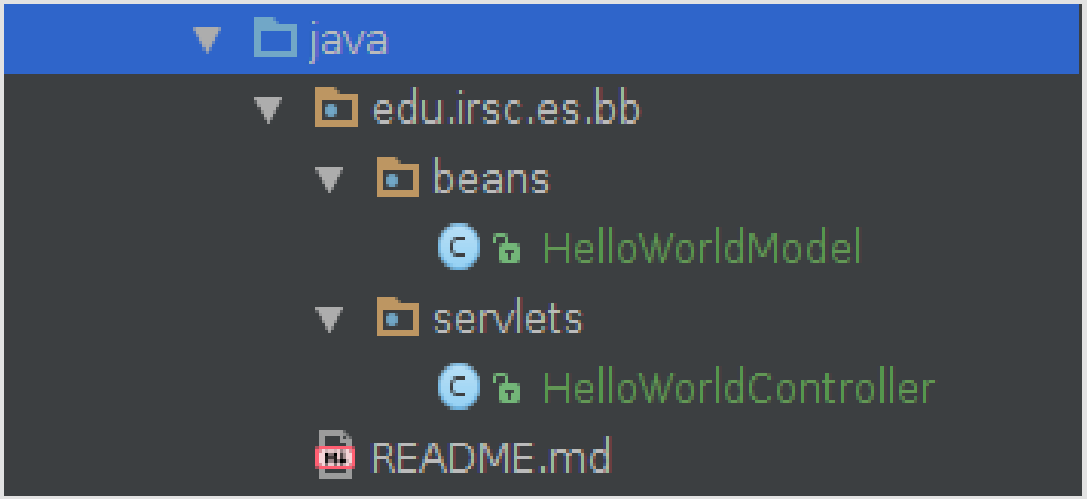
Hello World configure.jsp in BbUI



Expanding with Java MVC

Expanding with Java MVC

- Let's go back to project and make some changes.
- Create basic namespacing package in the src/main/java



For this example:
package setup - edu.irsc.es.bb.
Separate the models - (beans) class from the controllers - (servlets).

Creating a basic Model/Bean

- Create a class in the src/main/java/<package>/beans
- Copy and paste the following:

```
package edu.irsc.es.bb.beans;

public class HelloWorldModel {
    public String getHello(){
        return "Hello World!";
    }
}
```

This simple model/bean* is just a basic Java class for now, but this usually contains complex data manipulation and can also be used to control interaction with the database on a more secured/static method concept.

* Note: We purposely use two different terms to help relate what the function this section is to in the MVC architecture. In Java, this essentially is a Bean which translates to the Model in MVC.

Creating a basic Controller/Servlet

- Create a class in the src/main/java/<package>/servlets
- Copy and paste the following:

```
package edu.irsc.es.bb.servlets;
import edu.irsc.es.bb.beans.HelloWorldModel;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
public class HelloWorldController extends HttpServlet {
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doPost(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        HelloWorldModel hello = new HelloWorldModel();
        String hello_world = hello.getHello();
        request.setAttribute("hello", hello_world);
        RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/module/index.jsp");
        dispatcher.forward(request, response) ;
    }
}
```

Here is a basic servlet/controller* that has two override methods: doGet() and doPost() that act like a router for the particular url pattern caught. It performs conditional methods based on what is requested and then returns the computed data to a view that will be implemented.

*Note: as with the model/bean terms, we use the terms: servlets (Java) and controllers (MVC).
Servlets are the Controllers

Modifying index.jsp

- Open up the module/index.jsp – the view page.
- Add the following to the top of the jsp page:

```
<%@ page import="java.util.*,java.lang.*"%>
```

- Add the following code snippet below the taglib includes:

```
<%
    String hello = request.getAttribute("hello").toString();
%>
```

Modifying index.jsp

- bbNG:tabbedPanel content area change `${tabbed_panel}` to `${hello}`

- From:

```
<bbNG:tabbedPanels>
    <bbNG:tabbedPanel title="${tabbed_panel}">
        ${tabbed_panel}
    </bbNG:tabbedPanel>
</bbNG:tabbedPanels>
```

- To:

```
<bbNG:tabbedPanels>
    <bbNG:tabbedPanel title="${tabbed_panel}">
        ${hello}
    </bbNG:tabbedPanel>
</bbNG:tabbedPanels>
```

Registering the Servlet with Hello World

- Register servlet & bean with web app (B2|plugin)
- Remember web.xml?
  - Contains the workflow of the overall application
- Open **src/main/webapp/WEB-INF/web.xml**

Replace the web.xml inner nodes as follows

```
<display-name>Hello World Plugin</display-name>
<servlet>
  <description>Simple Hello World Example</description>
  <display-name>Hello</display-name>
  <servlet-name>Hello</servlet-name>
  <servlet-class>edu.irsc.es.bb.servlets.HelloWorldController</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>Hello</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

<servlet> node registers the servlet/controller so it can be used from within Bb. Next add a route or servlet-mapping to allow it to be used.

For this example, create a mapping to look for "**/hello**" relative to the plugins url:

http://localhost:9876/webapps/irsc-hello-world-BBLEARN**/hello**

Before deploying Hello World changes

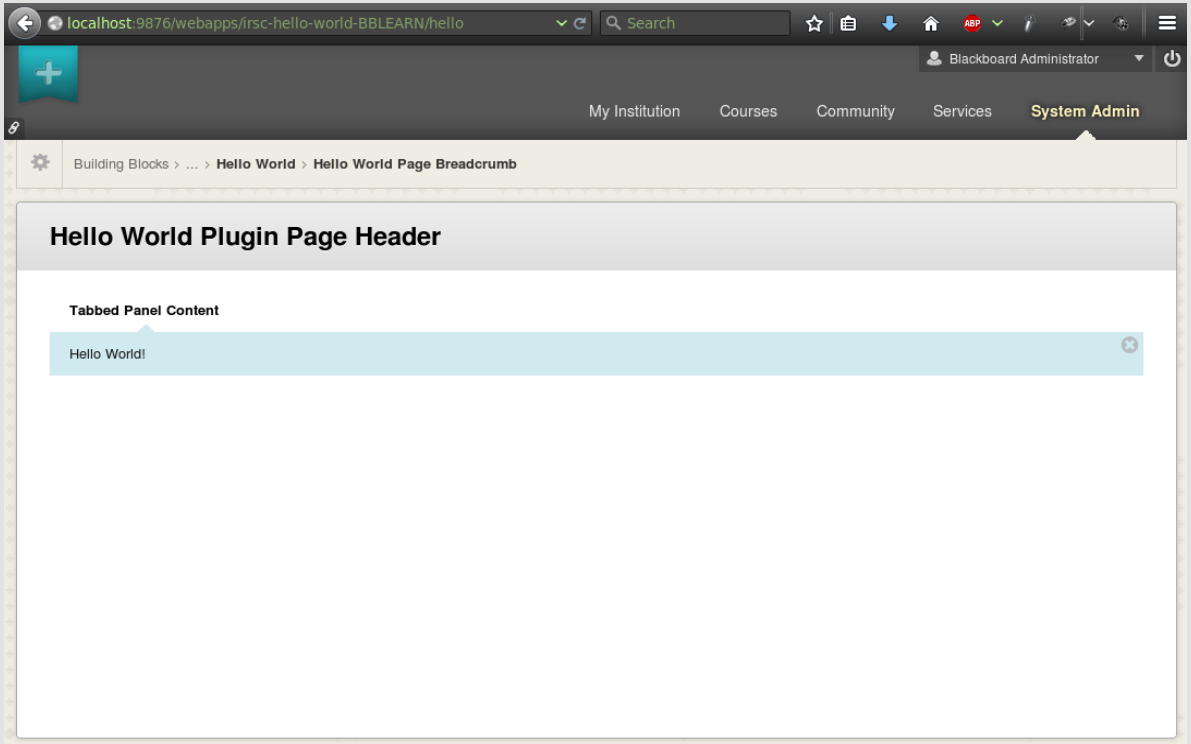
- Change line in bb-manifest.xml:
  - In the application-defs node
    - Links
      - Change value of url attribute to the mapping pattern set in web.xml

```
<link>
    <type value="system_tool"/>
    <name value="Hello World Tool"/>
    <url value="hello"/>
     <description value="app1.description"/>
</link>
```

This is relative to the application, and it will create the link into the Tools and Utilities with: app/path/hello

Deploy! ☺

- Run “gradle deployB2” command in terminal, etc.
- Note: See “Hello World!” in tabbed panel callout provided by HelloWorldModel bean imported from HelloWorldController servlet when accessed through route servlet mapping just created! ☺
- If receiving an error like: Unsupported major.minor version 52.0, then make sure current java_home or jdk reference is at least same version as Developer VM



Using Beans/Models Directly

- We can also use the HelloWorldModel directly

  Edit index.jsp and add:

```
<jsp:useBean id="HWM"
class="edu.irsc.es.bb.beans.HelloWorldModel"
scope="session"/>
```

  Placed just after the fmt:message section

  Next add a **<H4>** element with the Model.func() method:

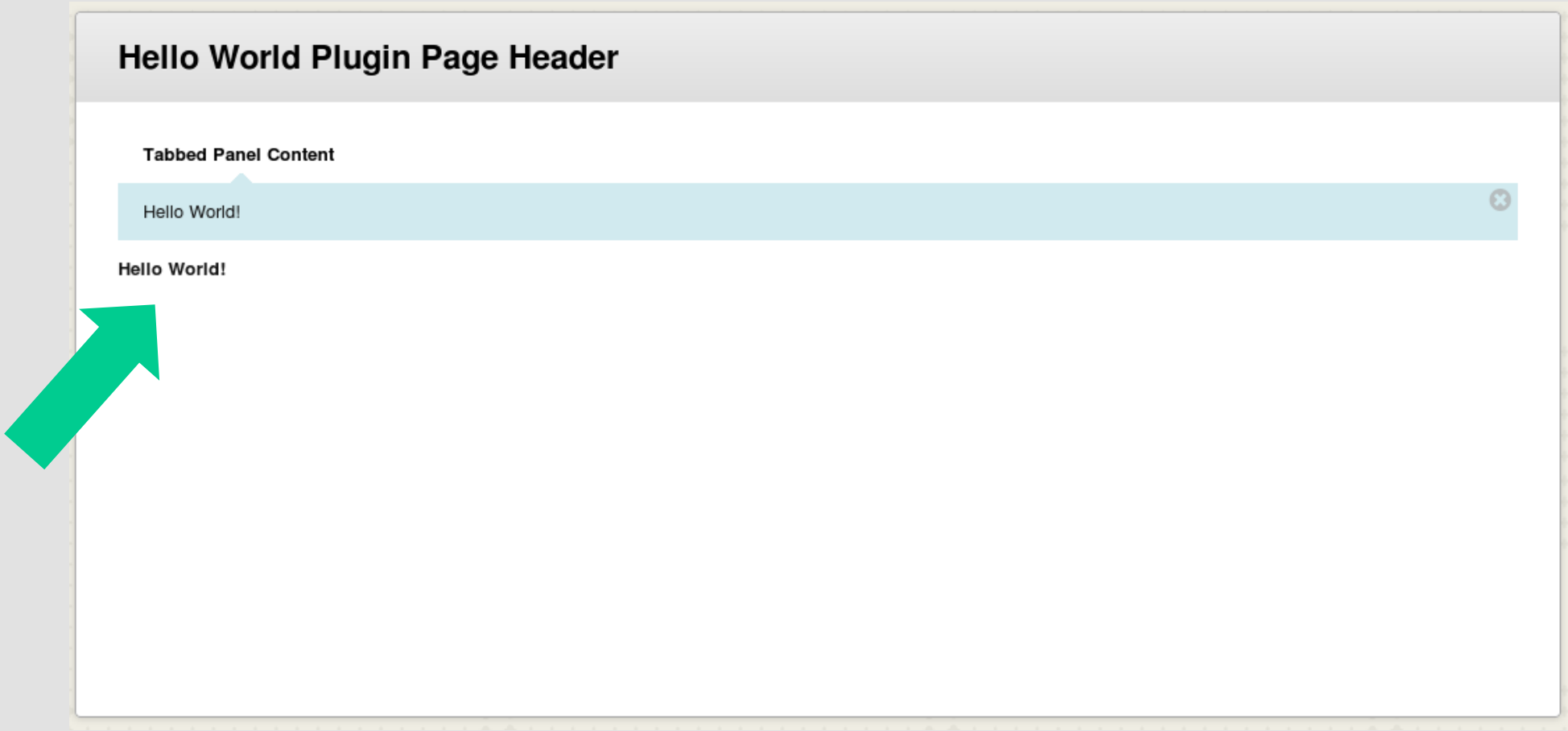
```
<h4>${HWM.getHello()}</h4>
```

Example index.jsp

```
<%@ page import="java.util.*,java.lang.*"%>
<%@ taglib uri="/bbData" prefix="bbData"%>
<%@ taglib uri="/bbNG" prefix="bbNG"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<%
    String hello = request.getAttribute("hello").toString();
%>
<bbData:context>
    <bbNG:genericPage>
        <fmt:setBundle basename="settings" />
        <fmt:message key="name" var="plugin_name"/>
        <fmt:message key="title" var="plugin_title"/>
        <fmt:message key="hw.main.page.breadcrumb" var="plugin_breadcrumb"/>
        <fmt:message key="hw.main.page.header" var="page_title"/>
        <fmt:message key="hw.main.tabbed.panel" var="tabbed_panel"/>
        <jsp:useBean id="HWM" class="edu.irsc.es.bb.beans.HelloWorldModel" scope="session"/>
        <bbNG:breadcrumbBar environment="SYS_ADMIN" navItem="admin_plugin_manage">
            <bbNG:breadcrumb href="${recallUrl}" title="${plugin_name}" />
            <bbNG:breadcrumb title="${plugin_breadcrumb}" />
        </bbNG:breadcrumbBar>
        <bbNG:pageHeader>
            <bbNG:pageTitleBar title="${page_title}">
                ${page_title}
            </bbNG:pageTitleBar>
        </bbNG:pageHeader>
        <bbNG:tabbedPanels>
            <bbNG:tabbedPanel title="${tabbed_panel}">
                ${hello}
            </bbNG:tabbedPanel>
        </bbNG:tabbedPanels>
        <h4>${HWM.getHello()}</h4>
    </bbNG:genericPage>
</bbData:context>
```

Result after Deploy ☺



THANK-YOU FOR ATTENDING

Be sure to visit us at:

http://www.irsc.edu

Indian River State College

3209 Virginia Ave

Fort Pierce, FL 34981