IBM Connections 4.0 – Social Software for Business

Lab Manual

IBM Connections Features Walk-through

Introduction:

IBM Connections is the market leading, integrated and secure social software platform that helps people engage with networks of expertise in the context of critical business processes in order to act with confidence and anticipate and respond to emerging opportunities.

IBM Connections is comprised of several different social applications. It's important to note that these applications can be used within an organization as needed — all together or as individual applications.

IBM Connections is social networking software that consists of these features: Activities, Blogs,

Bookmarks, Communities, Files, Profiles, Forums, Mobile, and Wikis.

You can use these IBM Connections features to accomplish the following goals:

- Activities - Collaboration tool for collecting, organizing, sharing, and reusing work that is related to a project goal.
- Blogs - Online journals that you can use to deliver timely information with a personal touch. You can use a blog to present your ideas and get feedback from others or learn from the expertise of others who blog.
- Bookmarks - Social bookmarking tool for saving, organizing, and sharing Internet and intranet bookmarks. Discover bookmarks that have been created by others with similar interests and expertise.
- Communities - A web site where people who share a common interest can interact with one another, share information, and exchange ideas. Community members can participate in community-specific activities and discussion forums, and can share blogs, bookmarks, feeds, and files.
- Files - A common repository in which you can store files and share them with others. Store versions of a file, view who has downloaded a file or commented on it, and see file ratings.
- Home page - A single Web page that helps you manage your attention by providing a glimpse at relevant social data collected from across your subscriptions, notifications, and network of colleagues.
- Profiles - Directory of the people in your organization, including the information you need to form and encourage effective networks across your organization.
- Forums - An online discussion board where people can ask questions, share their experiences, and discuss topics of common interest.
- Wikis - A tool for creating wikis that individuals, groups, and communities can use to capture, share, and co-author information. View page changes, ratings, and comments.

Together, these tools will help you and your colleagues interact with one another more effectively.

Description:

The purpose of this lab is to take you through a walk-through, or demonstration, of the various new features of IBM Connections 4.0. Developer-specific features will be covered in depth with succeeding labs in this workshop. Throughout the lab you will be logging with several users. This helps facilitates the demonstration of the features within Connections.

Objective:

At the end of the lab, you should be able to:

- Have a better understanding of the new features provided with IBM Connections
- Feel more comfortable with the ease of use of IBM Connections
- Become better prepared for the additional labs that provide more depth

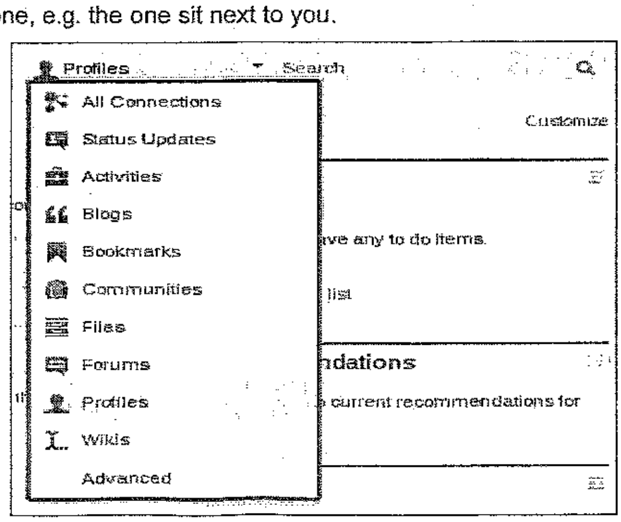
Procedure:

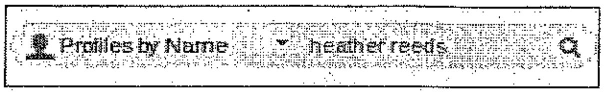
INVITE-TO-MY NETWORK

As collaborative social software IBM Connections enables users to connect with each other in various ways. Networking with other users is fundamental in the overall effectiveness of Connections, and something that provides great value to any Connections user. In this section we walk through how a user can expand their network via a user's profile page.

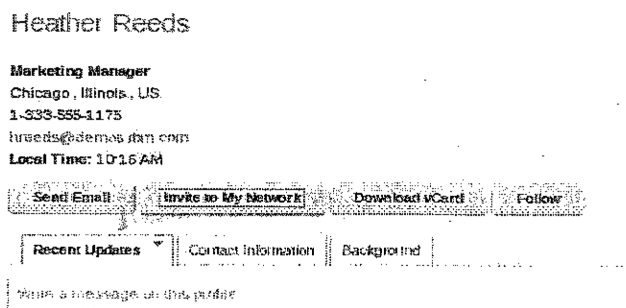
Step 1 Open a browser and navigate to the Connections homepage **https://greenhouse.lotus.com/homepage/**

Step 2 Logon using your given **user name** and **password**. Navigate to a user's profile that you would like to network with. In the upper right corner select **Profiles** from the drop down list. Search for someone, e.g. the one sit next to you.

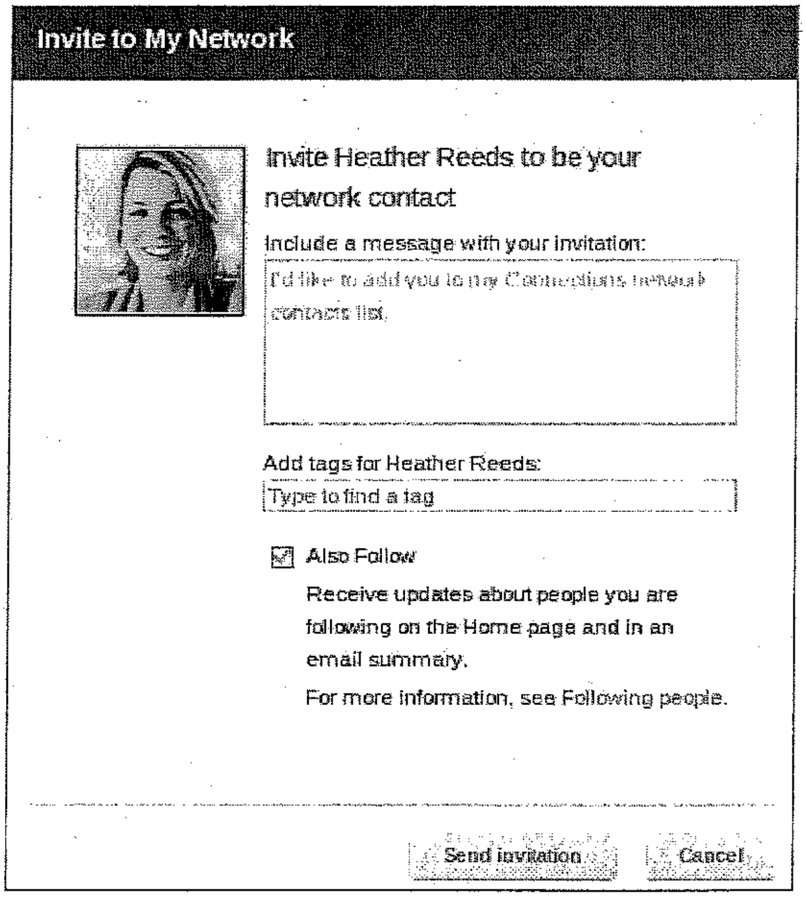




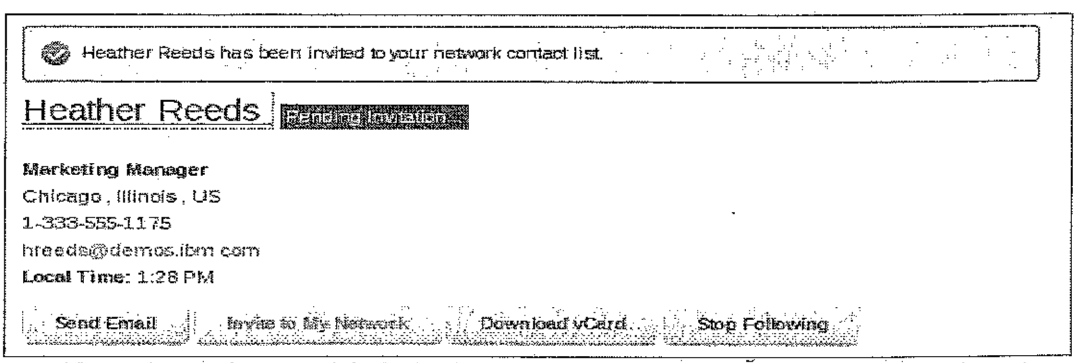
Step 3 When the Profile Search Results is displayed, click on his/her name to display his/her Profile. Click on the **Invite to My Network** button, located above the user's activity stream on the profile page.



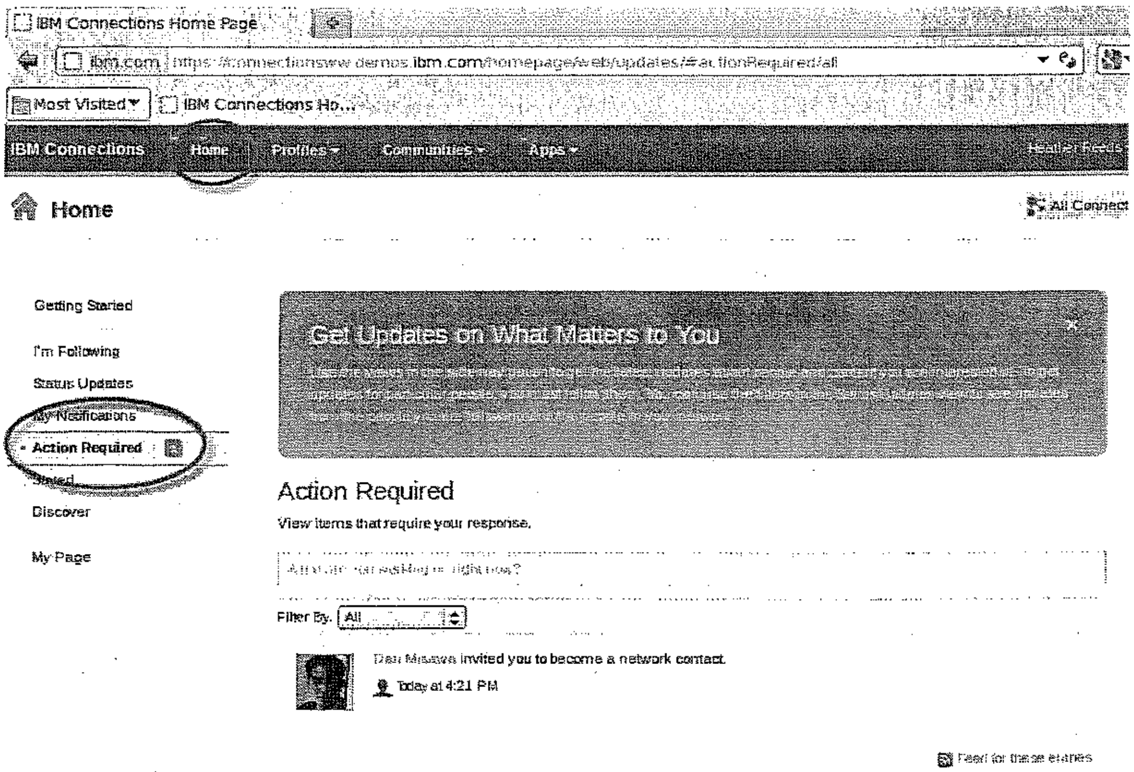
Step 4 This will launch the Invite to My Network window. By default you have a standard message the recipient will see upon receiving this network invitation. You have the option to customize this message and also add a tag to associate with this user once in your network. The last option in the window will allow you to follow the user in addition to adding them to your network. By default the follow option is enabled. Make your selections and click **Send invitation**.



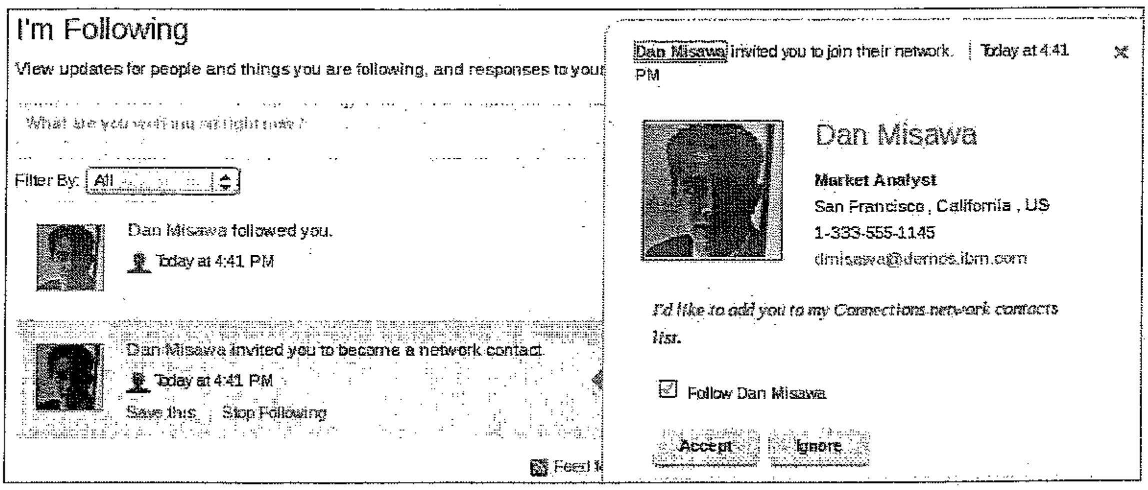
Step 5 Once the invitation is sent you will see a confirmation message above the activity stream. You will also note a pending invitation message on the user's profile page.



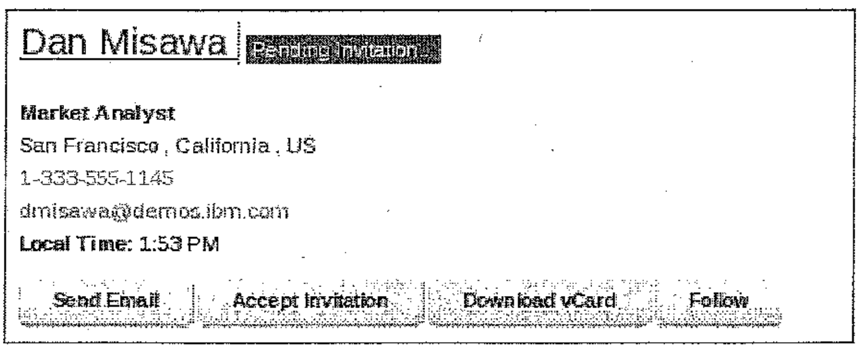
Step 6 The recipient of a network invitation has 2 ways of accepting. The first way is to navigate to your **Actions Required** section from your **Home** page.



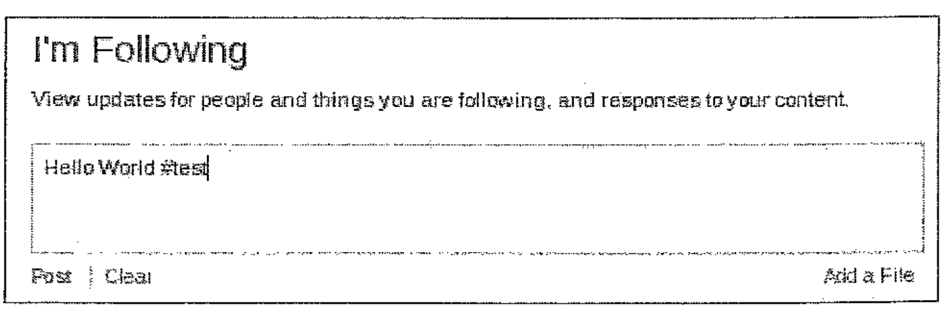
Step 7 Here you sill see a note in the left hand menu calling your attention to a new network invitation. Clicking inside the network invitation itself (click on >> within the highlighted invitation) will show you an embedded windows where you can Accept the invitation. You'll receive a confirmation message indicating that you are now network contacts and following someone, e.g. Dan Misawa.



Step 8 Alternatively, you can access the user's profile that sent you the invitation (for example by clicking on their name) (Hint: Search on e.g. Dan Misawa to find his Profile then click on his profile to display more information) and confirm the network invitation from their profile page by clicking the **Accept Invitation** button.



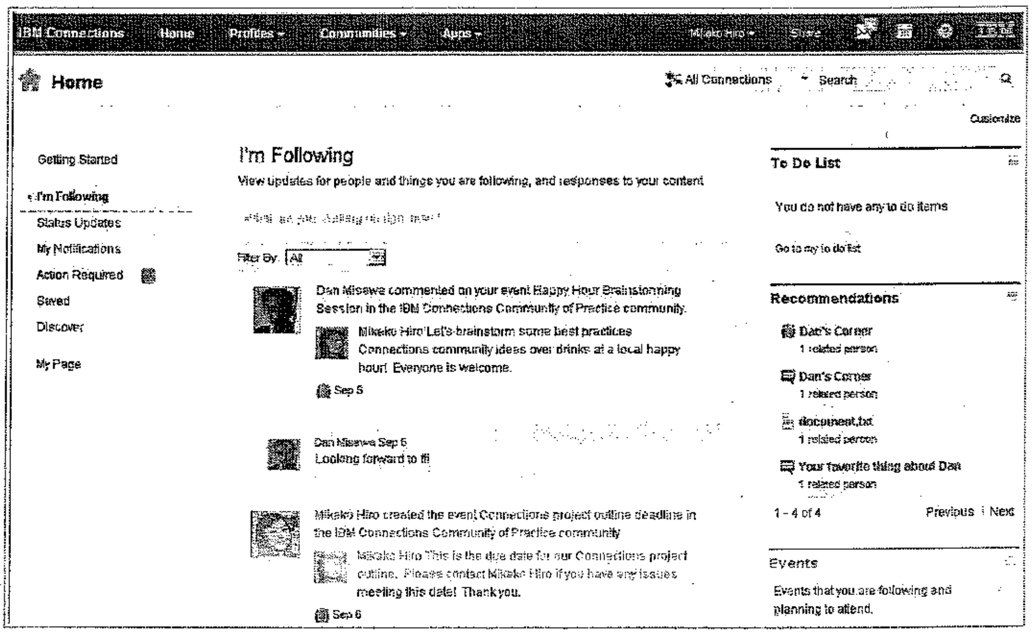
Step 9 While logged on, under the **I'm Following** header you will see a text box with the place holder text "What are you working on right now?". Click inside this text box and enter a simple status message (Hello World #test), then click Post. You'll see later how this shows up in others Activity Streams. Click **I'm Following** from Connections **Home**. You will now see that the invitation status in your activity stream that was accepted.



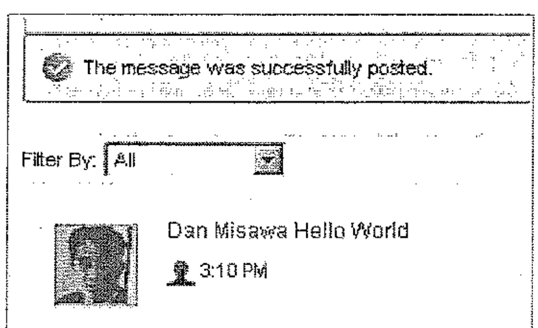
Status Updates, hash tags, and the Share Box

Step 1 In this section you will explore interaction with the Connections activity stream. From your homepage you can explore various ways to interact and share with users, including the new Connections share box.

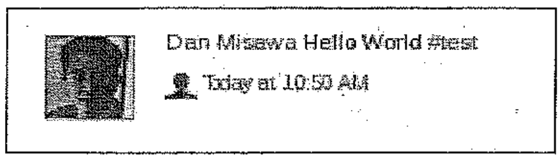
Step 2 Logon and navigate to Connections **Home** page. Content on the page may vary depending on activity already done.



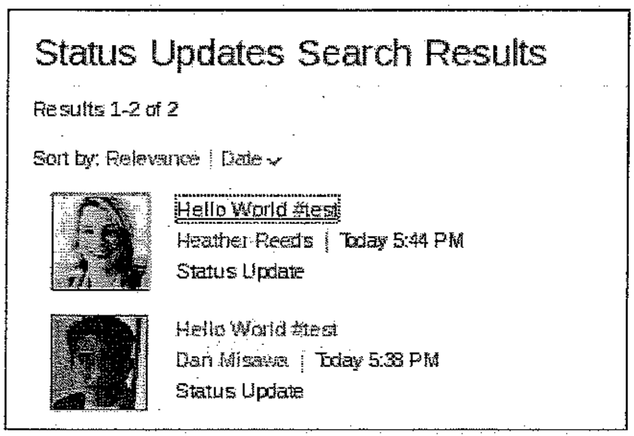
Step 3 Try a simple status post. Under the I'm Following header you will see a text box with the place holder text "What are you working on right now?". Click inside this text box and enter a simple status message (**Hello World**), then click Post.



Step 4 You can also utilize hash tags as a new feature of Connections. Try doing another status update, but this time end the update with a hash tag (**#test**).

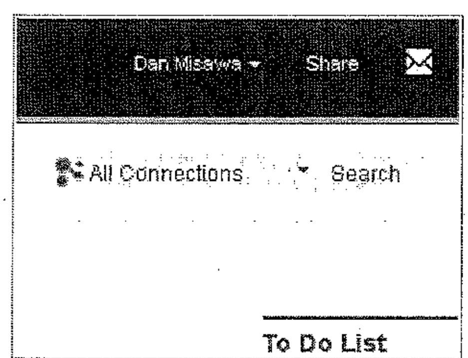


Step 5 After posting your new status update click on the hash tag. This feature will search information throughout Connections that you have access to and display search results for all items that contain the hash tag **test**. As a suggestion, try using this feature at the end of the lab, since there will be more content added throughout the lab using the hash tag of **#test**.

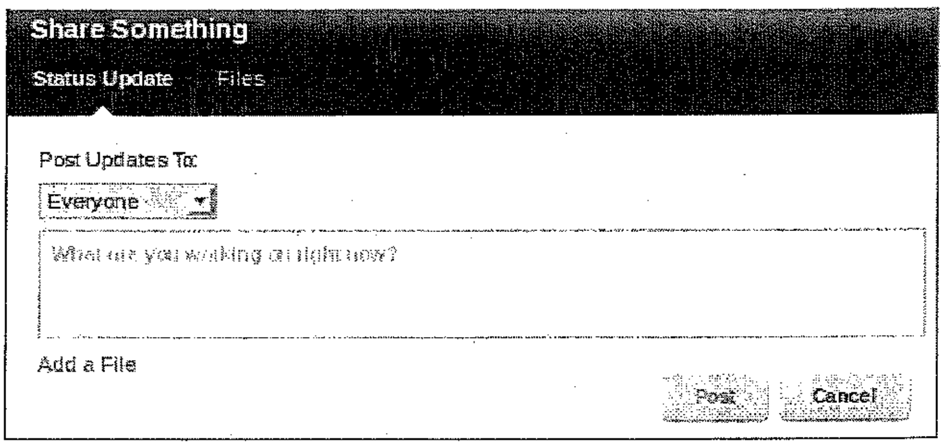


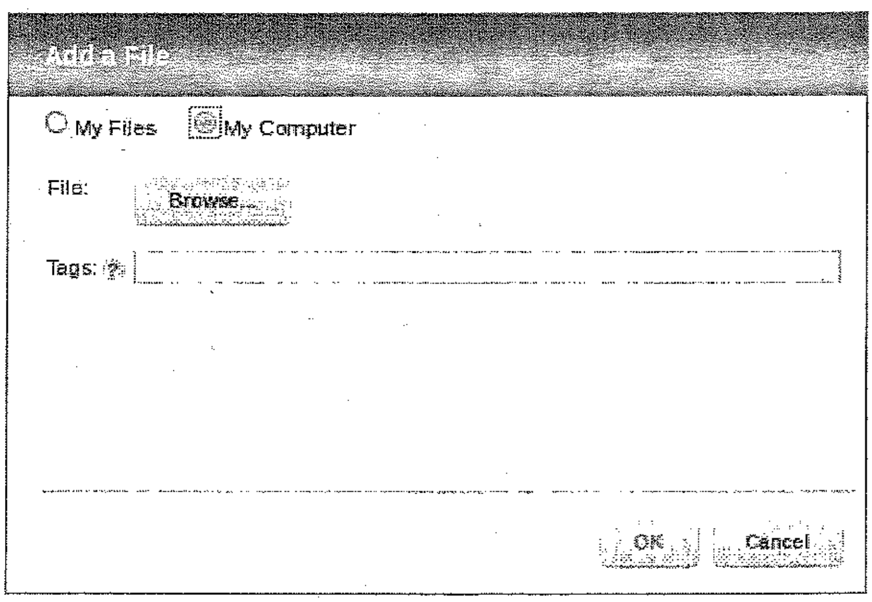
Step 6 The Share box is a new feature to Connections. The Share box is accessible from any page so you always have a quick and convenient way to share information and files with others.

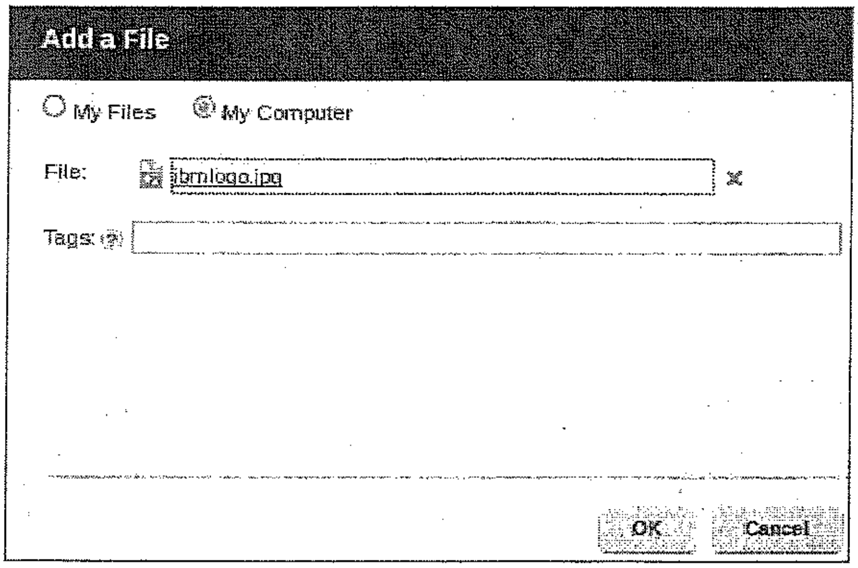
Step 7 You can also share status information and files through the Share box. Click on the **Share** box, located in the top right section of your screen. You should then see the **Share Something** window open.



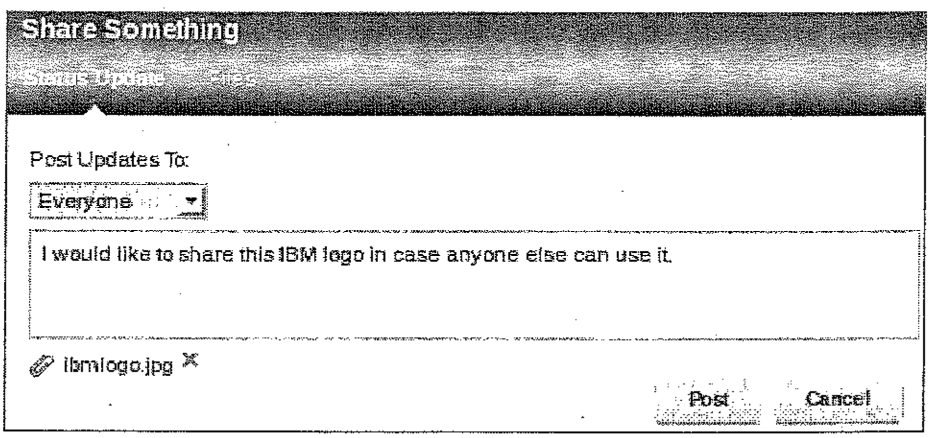
Step 8 Now in the Share Something window you have a few options. You can post through the Status Update (as previously done), you can post using Files, or you can post through Status Update and include files from either your local computer or from within Connections. Here we will share a file from your computer into your Connections account, and also give some context by including a status message. Click on the Add a File link underneath the Status Update text box, and via the file browser navigate to the **Lab_helper_files** folder and select the **ibmlogo.jpg** file to be uploaded. (Note: The IBM logo graphic file was used for illustration purpose. You could post any file that is small and easily selectable.) Then click OK.



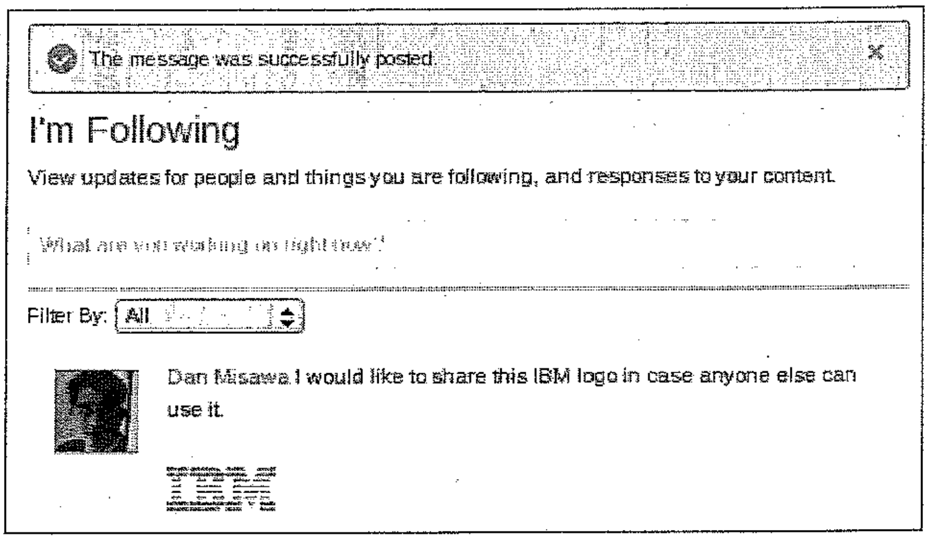




Step 9 Once you are back on the **Share Something** window you should add a status pertaining to your file you are about to post. Once you have included a status message click the **Post** button.



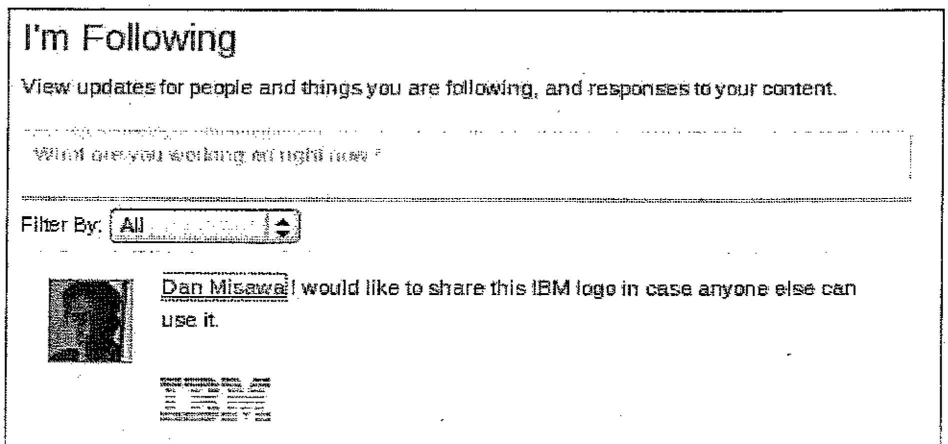
Step 10 Click on **Home** then **I'm Following.** You should now see a confirmation message in addition to the status update and picture displayed in your activity stream.


The message was successfully posted.
I'm Following
View updates for people and things you are following, and responses to your content.
What are you working on right now?
Filter By: All
Dan Misawa I would like to share this IBM logo in case anyone else can use it.

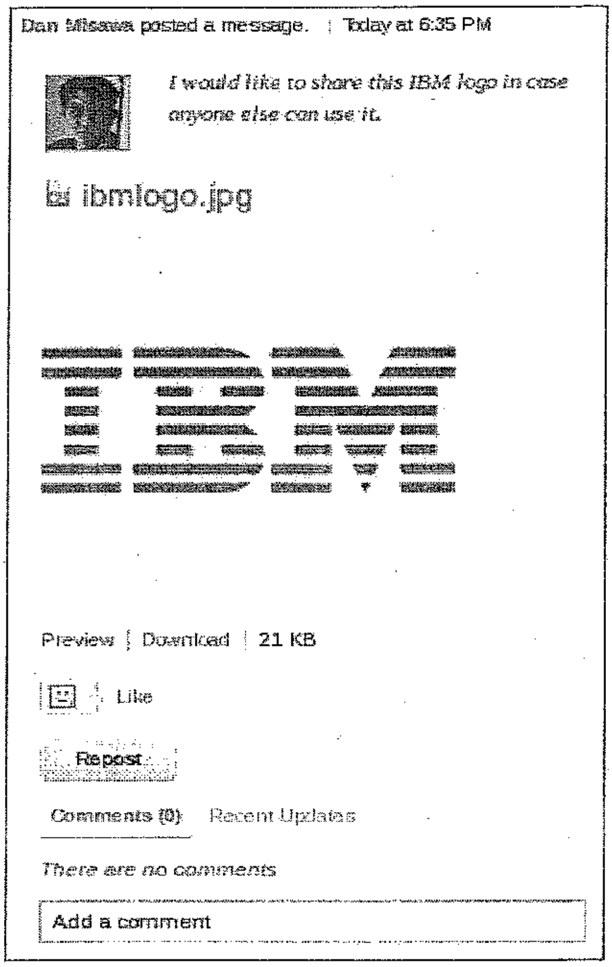
Embedded Window Features

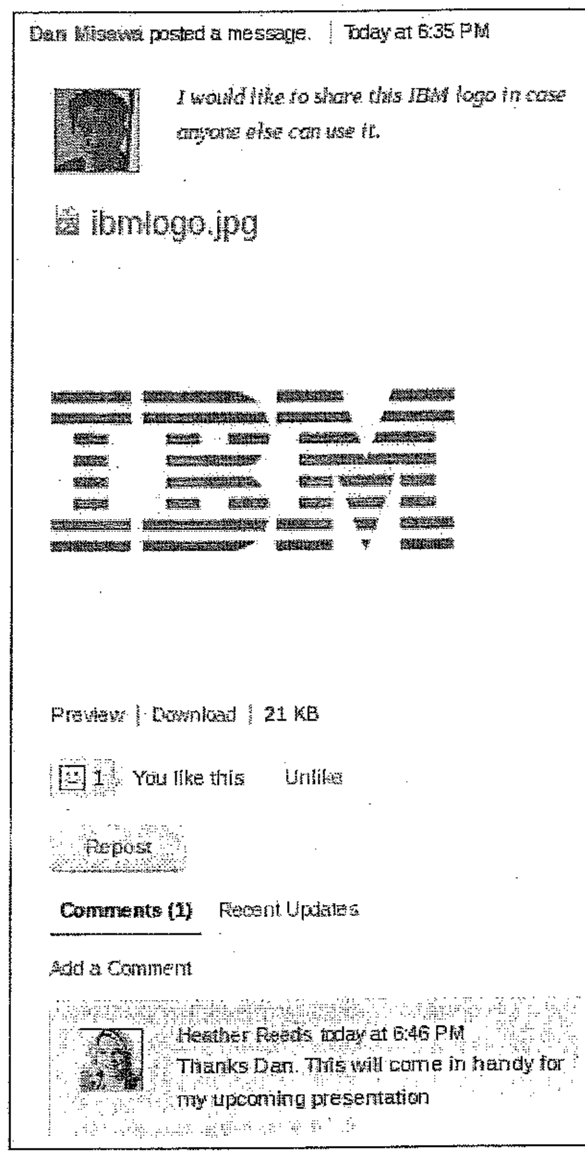
The embedded experience window is a new feature of IBM Connections. You can utilize this new feature to quickly view and interact with various content without leaving your current page.

Step 1 Locate a recent post from someone you are following, displayed in your activity stream, the one that e.g. Dan Misawa just posted with the IBM logo. Notice when you hover your mouse over the post, the post is highlighted as blue, and you see right-facing arrows on the right side of the post. Click on these arrows to open the embedded window feature. Within the embedded window you will see a number of options to interact with the person and the update.



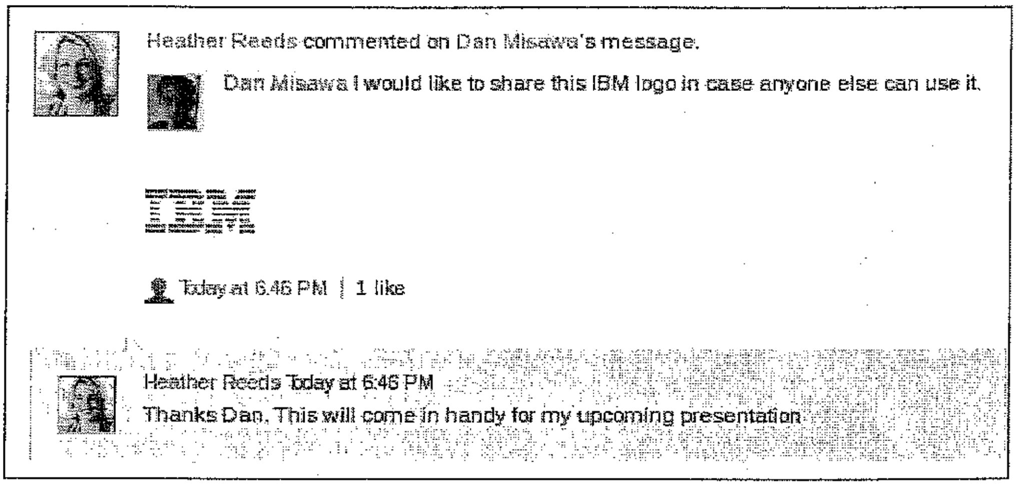
Step 2 Through this embedded window you can view the person's profile, repost the update, preview the attached file, download the attached file, 'like' the post, comment on the update, and view recent updates that have been made. Go ahead and interact with this update by clicking **Like** to provide feedback that you are interested in the content. In addition, enter a message into the **Comments** section (i.e. **Thanks Dan. This will come in handy for my upcoming presentation.**) and then click **Post**.


Dan Misawa posted a message. | Today at 6:35 PM
I would like to share this IBM logo in case anyone else can use it.
ibmlogo.jpg
IBM
Preview | Download | 21 KB
Like
Repost
Comments (0) Recent Updates
There are no comments
Add a comment


Dan Misawa posted a message. | Today at 6:35 PM
I would like to share this IBM logo in case anyone else can use it.
ibmlogo.jpg
IBM
Preview | Download | 21 KB
1 You like this Unlike
Repost
Comments (1) Recent Updates
Add a Comment
Heather Reeds today at 6:46 PM
Thanks Dan. This will come in handy for my upcoming presentation

Step 3 You can also try clicking on the **Download** link located under the attached file. Upon clicking you will be given the option to open the file with a default program, or save the file to your local machine.

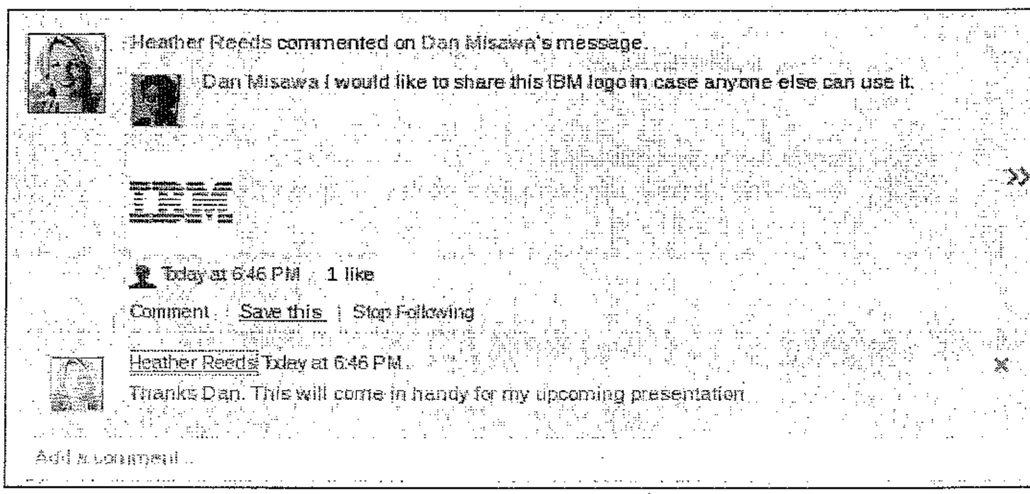
Step 4 You can close the embedded window by clicking the 'X' located in the top right area of the embedded window. You should now notice your recent interaction posted to the activity stream.



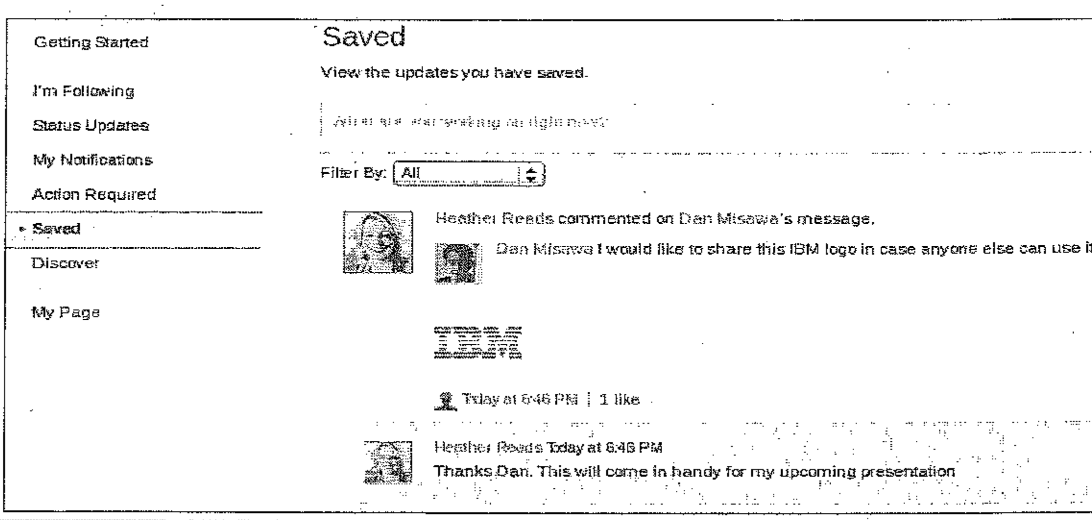
Save and Delet Items in the Activity Stream

The Connections activity stream allows users to save or delete items from their activity stream. This is useful to help users organize and retain important items in the activity stream which can be quickly accessed in the future. Connections also provides the ability to delete activity stream items a user may not wish to see.

Step 1 Locate a post you want to save, hover over with the mouse and click "**Save this**" to save the post in your **Saved** view. Note the "Save this" link has now changed to say "Saved", providing feedback that you have successfully saved this post.

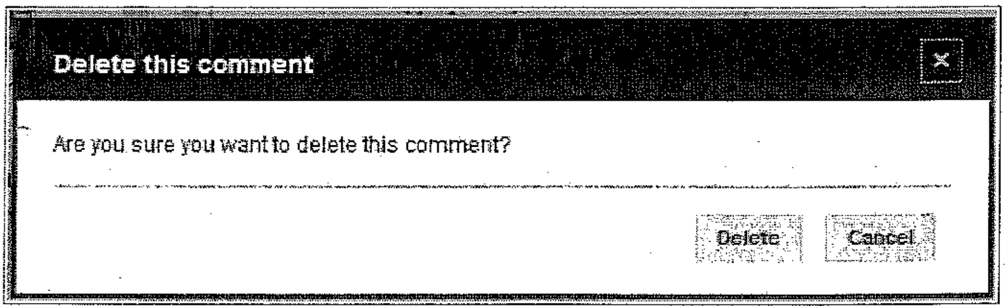


Step 2 In the left navigation menu click the Saved view and this will now display all saved postings.

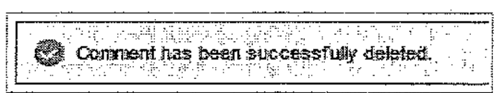


Step 3 Navigate to the **I'm Following** section in your left navigation menu. Locate a post that shows another user interacting with one of your activities (Posts may vary depending on others updates). In this case, you can delete any of your own posts). For example, this could be a post shown in your "Status Updates" section where another user has "Liked" something you previously posted. When you hover over updates of this type in your activity stream you will notice there is an optional '**X**' in the top right hand corner of the posting.

Step 4 You can delete a post by clicking the '**X**'. You will then see the Delete this comment confirmation window. You can click the Delete button to confirm.



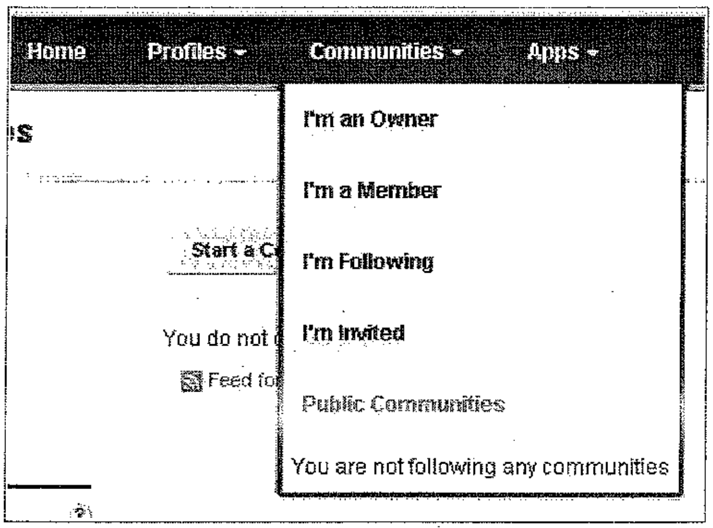
Step 5 you will notice the message is now gone from your activity stream and you see a confirmation message above the activity stream.



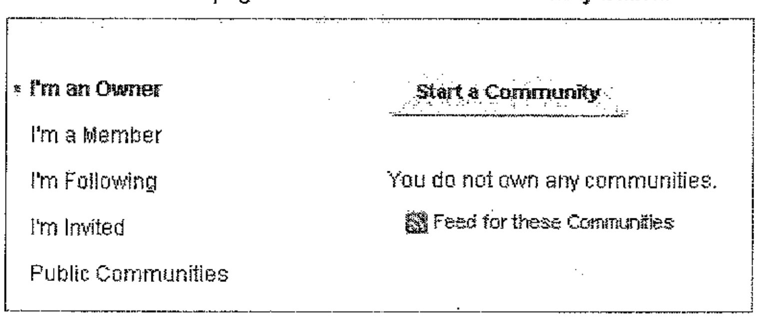
Creating a Community and Sharing Content

In this section, you will be introduced to the IBM Connections Communities feature. Communities are groups of people with a common interest. A community provides the means for users to stay in touch, share information, and exchange ideas. Communities provide an excellent way to connect members of a project team, organize a task force researching an emerging technology, or bring together a group of people who share any interest.

Step 1 To access your Communities area of Connections you can (from any page) mouse over the Communities drop down menu at the top of your screen, and select any of the navigation options below.

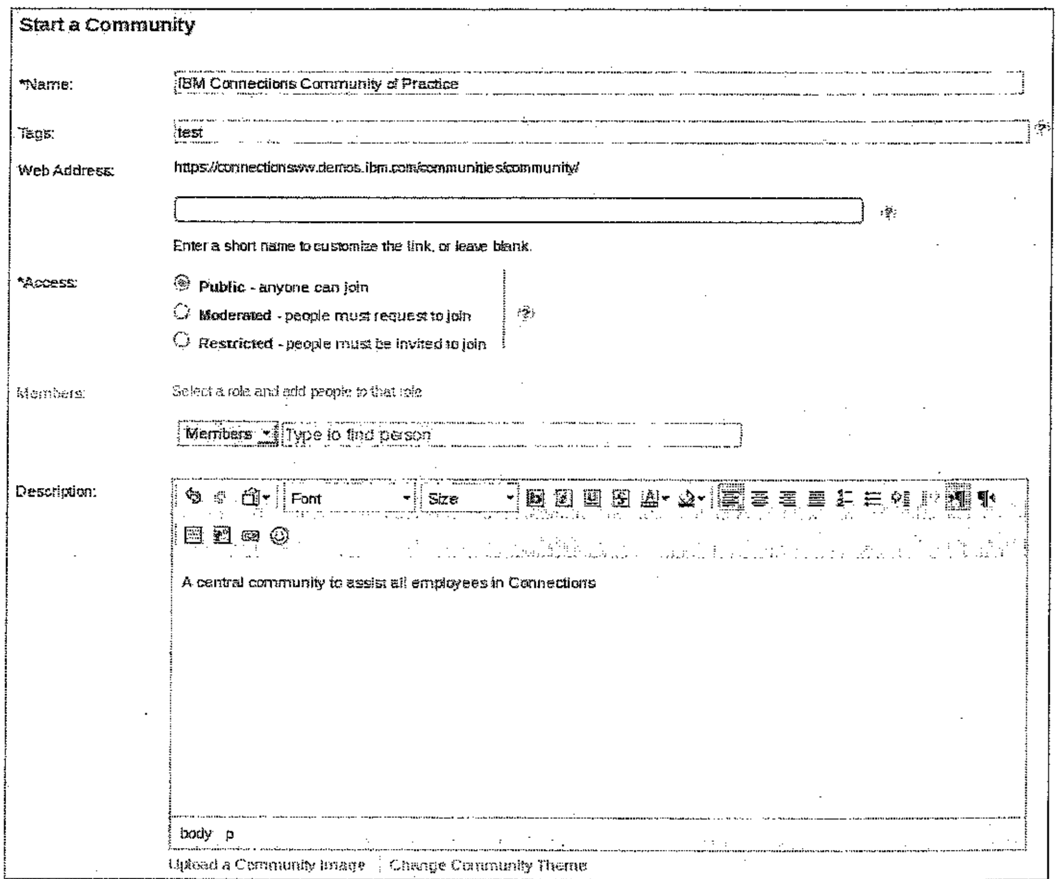


Step 2 Once on the **Communities** page click on the **Start a Community** button.

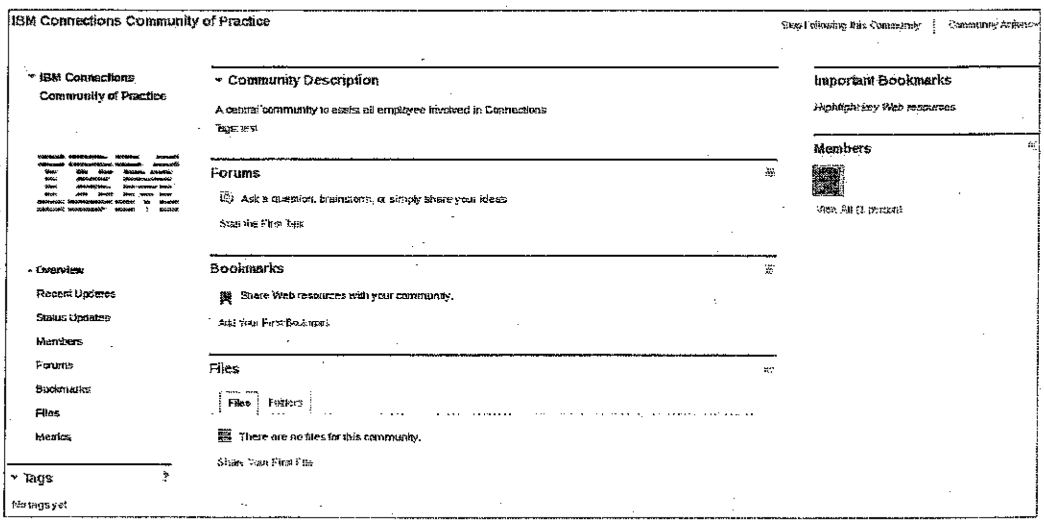


Step 3 You are now at the **Start a Community page**. Here you must fill out the required fields such as Name of the community and the type of access. You can also add optional data such as tags you want to associate with your community, and set up a list of people who can have various authorities to edit or manage your communities. It is also a good idea to provide a description of the community so users understand what the focus of this community will be. Below the description text box you will see a few options to Change Community Theme (where you can pick from a number of existing color themes) and also an area to upload your own picture file as the image associated with your community. Click on **Upload a Community Image**. Using the file browser, navigate to the **Lab_helper_files** folder and select the **ibmlogo.jpg** file. This image will also appear on your homepage.

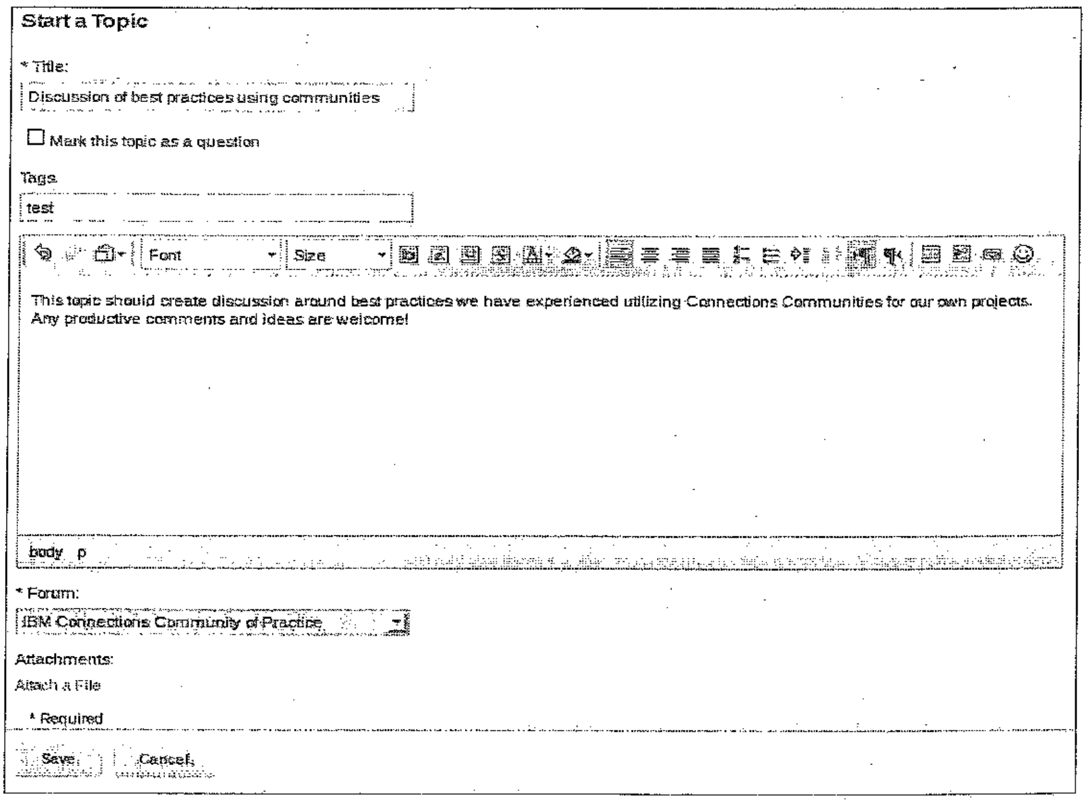
Once you are done filling out the Start Community page click the **Save** button.



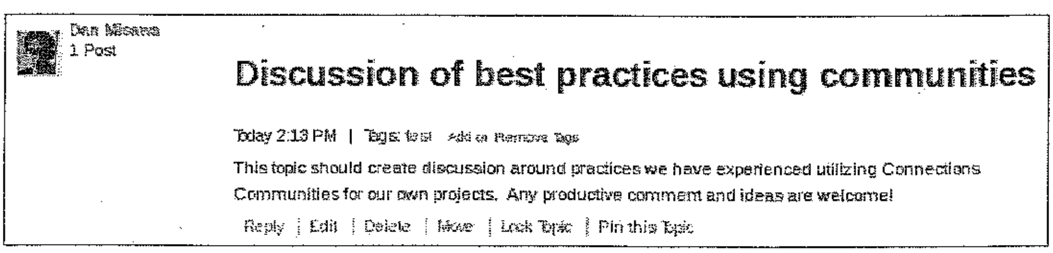
Step 4 After saving your selections you will be at your community homepage. Now you can start populating your community with useful information. We will start by adding content to the basic components of a community, the Forums, Bookmarks, and Files. These are located down the center of your community page. Locate the Forums section and click **Start the First Topic** to create a forum topic.



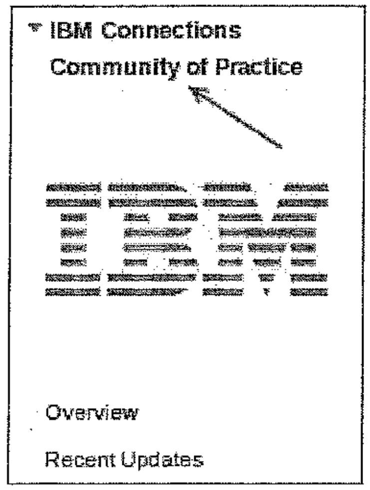
Step 5 You will now see the **Start a Topic** creation page. Fill out this page giving the forum title, tags, and description. Note you also have the option to attach a file which is located on your local computer. Once you are finished click the **Save** button.



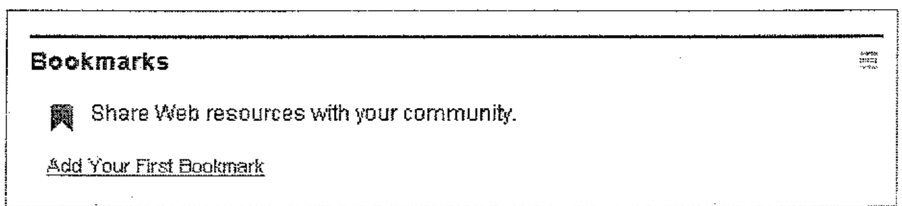
Step 6 You have just created the first Forum topic. As a member of the community you have several options to interact with this Forum post from the options listed directly below this post. You have the option to reply to this post, edit the information (if you have authority), move this post, lock the topic so no one else can alter or add to it, and pin this topic so you have a quick and convenient way to find this information.



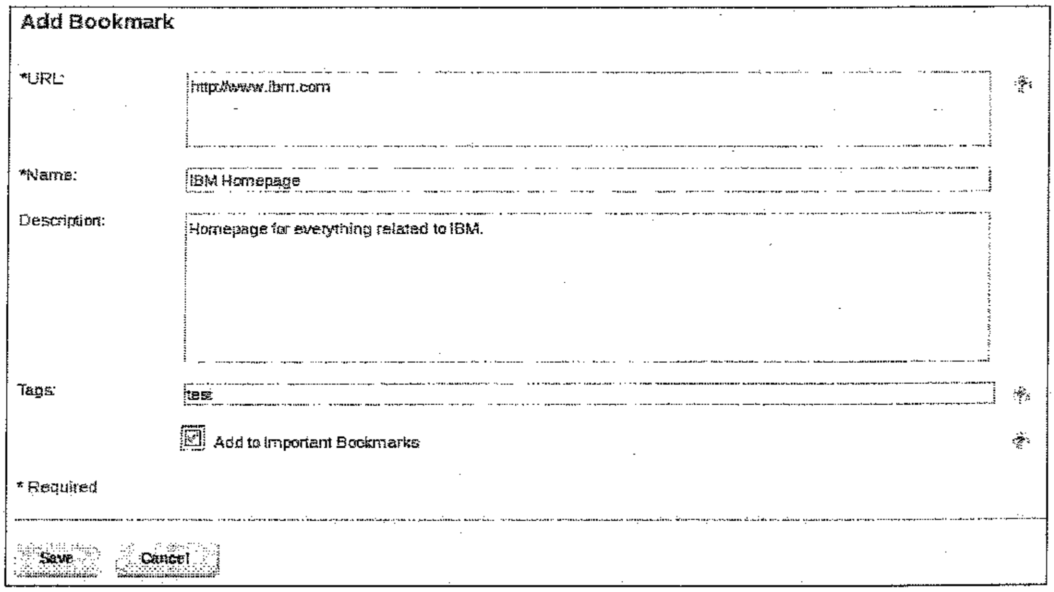
Step 7 Select the Community you created, to navigate back to the community homepage. Click on the name of the community above the Community's logo on the left hand side.



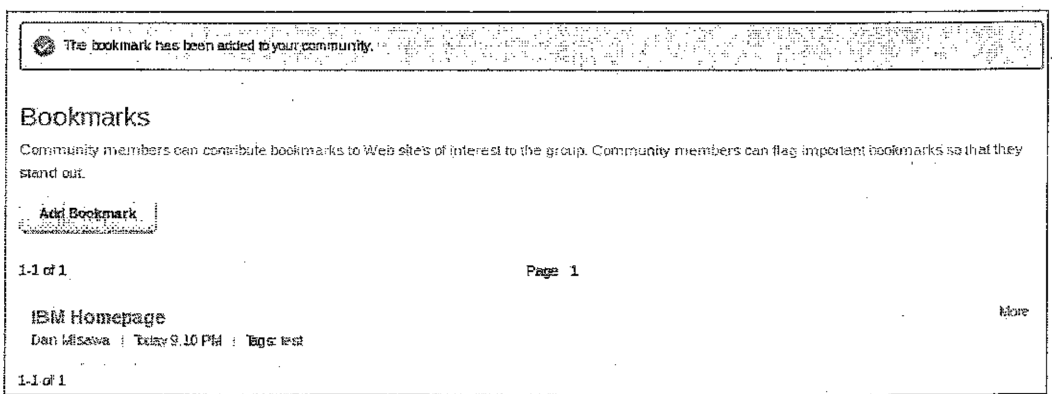
Step 8 Next, add a bookmark to your community by locating the Bookmarks section, located below the Forums section. Click the **Add Your First Bookmark** link to get started creating a community bookmark.



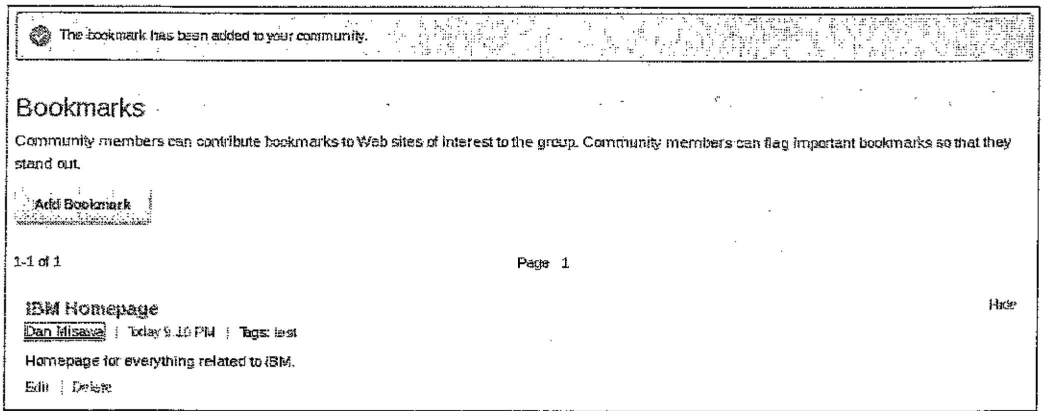
Step 9 You will see the Add Bookmark page. Fill out this page, including the URL, name, description, and any tags you wish to associate with your new community bookmark. When done click the **Save** button.



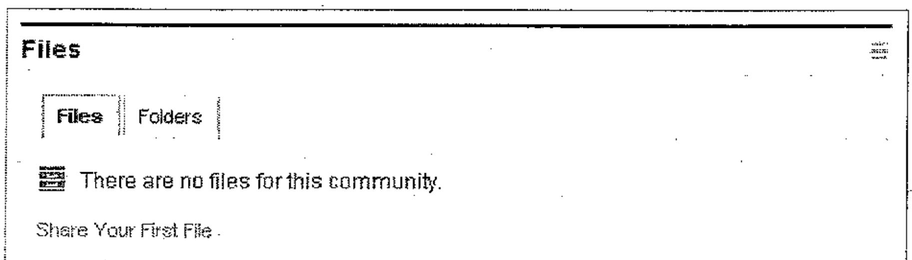
Step 10 Once the bookmark has been added to your community you will see a confirmation message at the top of your screen.



Step 11 Clicking on the **More** link to the right of your bookmark entry will provide you with the ability to **Edit** or **Delete** the bookmark you just created.



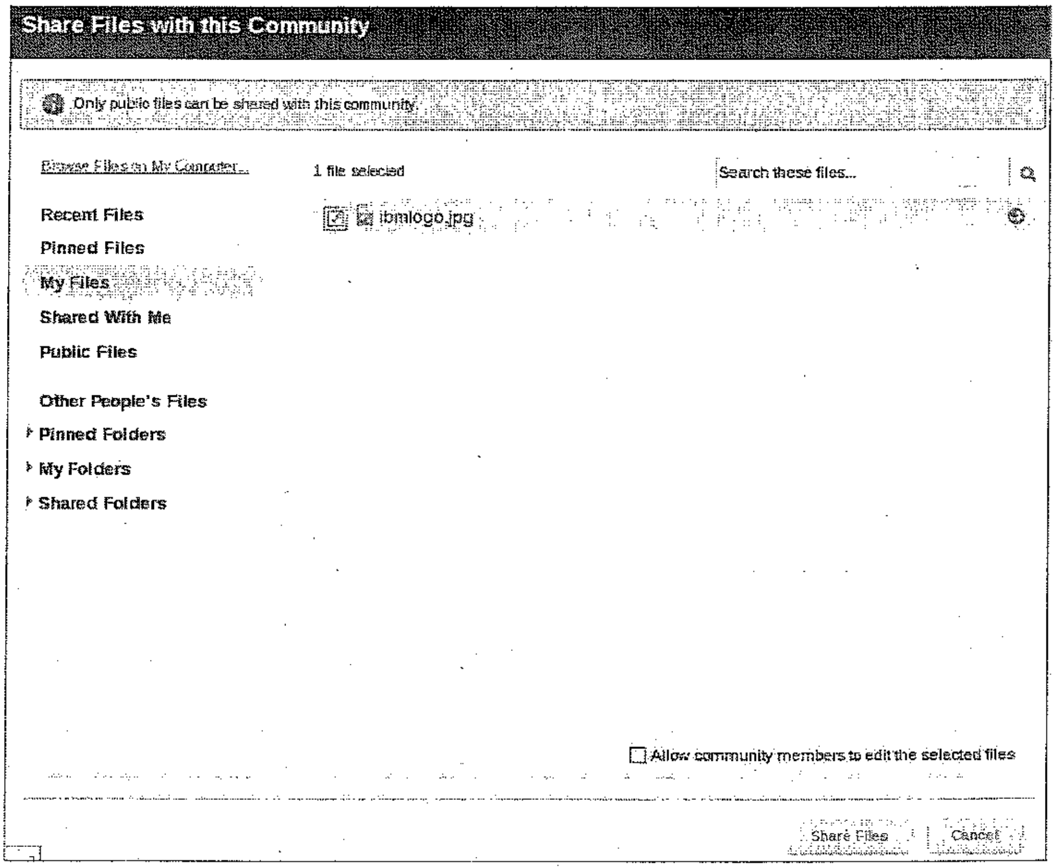
Step 12 Navigate back to the home page of the **IBM Connections Community of Practice** community. To add a file to your community locate the **Files** section of your community homepage, which is under the **Bookmarks** section. You will have to click Click the **Share Your First File**.



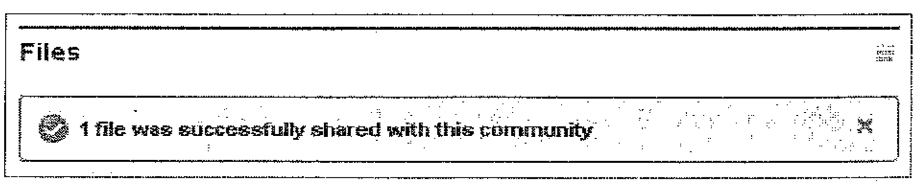
Step 13 You will now see the **Share Files with this Community** window. You have several options for sharing a file with this community. The first option you will see on your left menu will be to **Browse Files on My Computer**, which allows you to select a local file to upload. **Recent Files** will show you a list of files you have most recently interacted with. **Pinned Files** show any files you decided to pin through connections for quick reference. Using the **My Files** view will provide a list of the files you have uploaded yourself into connections. The **Shared With Me** option allows you to view files that other users have marked as sharing with you either through editor or reader access.

Other People's Files option allows you to search users for public access files you can incorporate into the community. **Pinned Folders** is similar to the previous pinned files option, as this view shows any folders you have pinned. The **My Folders** option shows you the folder structure you have created or uploaded for your specific user in Connections. Finally, the **Shared Folders** view is a list of all folders other users have marked as sharing with you.

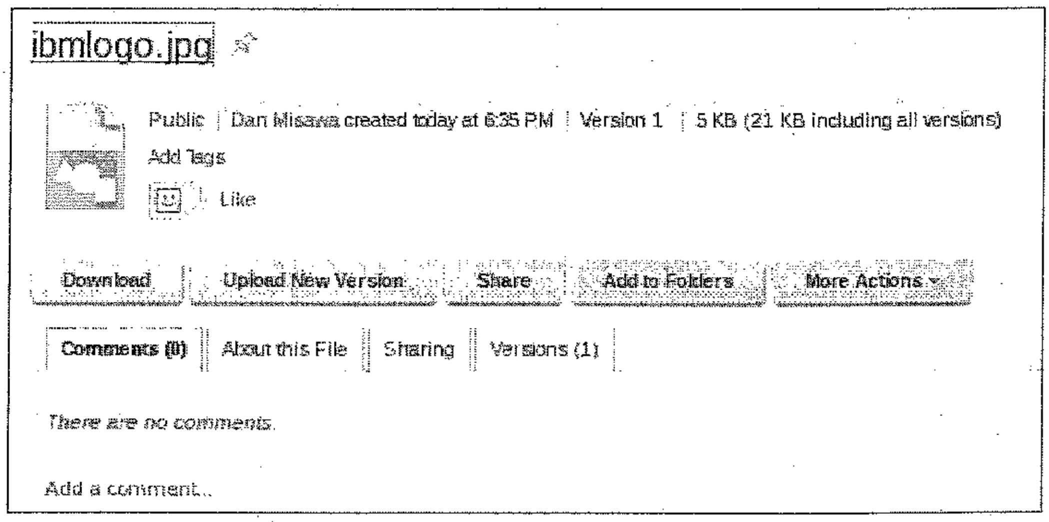
For the purposes of this lab we will click the **My Files** view and share a file this user has already uploaded to Connections (This was done during the earlier part of the lab. That file is **ibmlog.jpg.** should be displayed in the list. Check the file to share and then click **Share Files**.



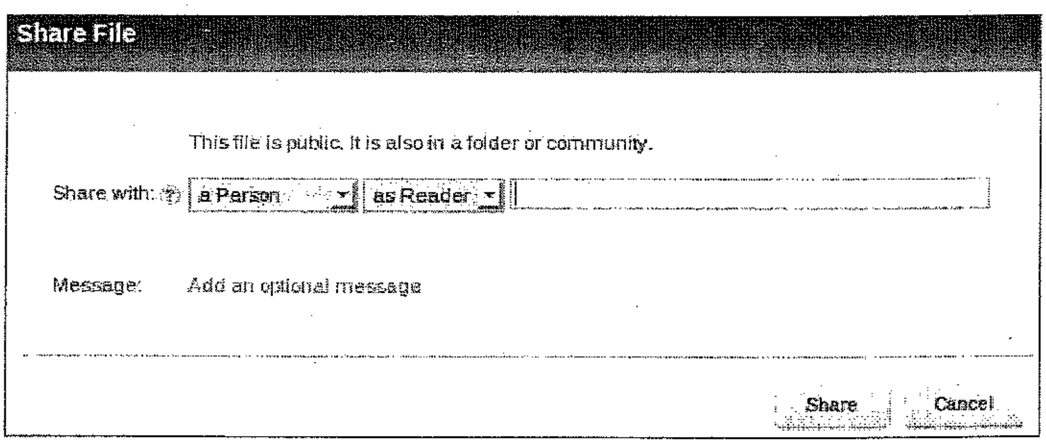
Step 14 Once your files has been added to the community you will see a confirmation message under the **Files** section.



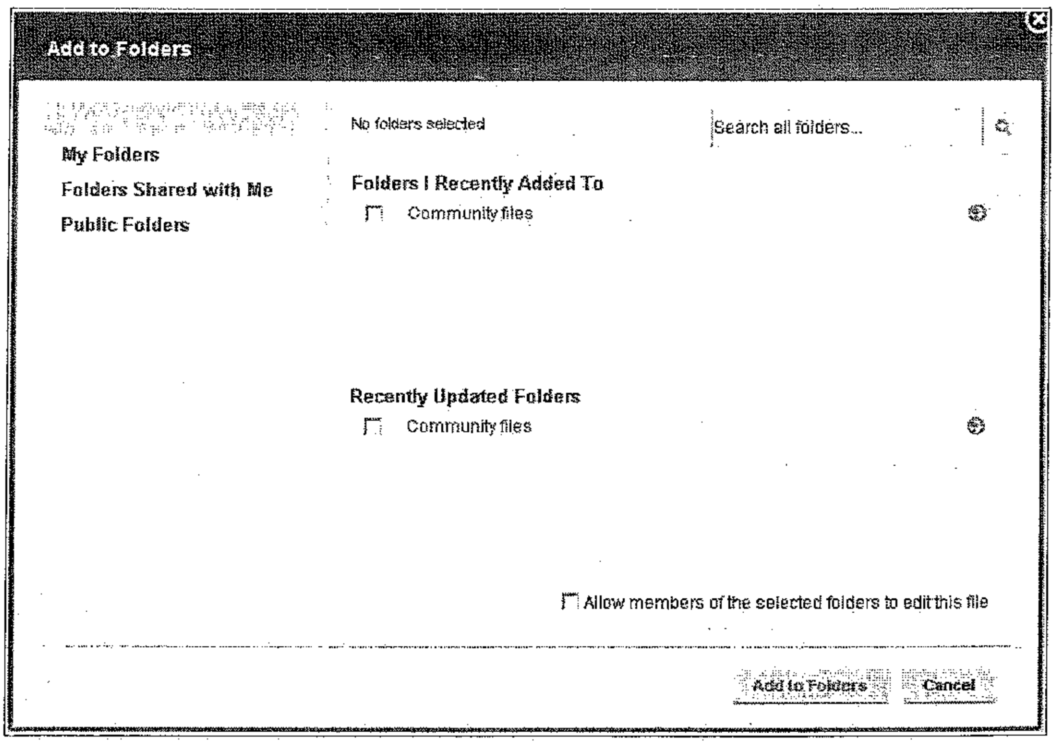
Step 15 Clicking on one of the file names within the **Files** section, will navigate you to the detailed page for this specific file. Once on this page users have a number if ways to interact with the specific file. At the top of the page near the file name note that you can 'pin' the file, which will cause this file to be listed under your **Pinned Files** view so you can quickly find this content at a later date. Users can click the **Like** link to tag the file and show other users this is something they are interested in. Along the middle note there are five buttons that allow you to manage and control the file. The **Download** button allows a user to download the file to their local computer. The **Upload New Version** button will launch a window allowing the user to select a file from the local computer to upload as the newest version of the file.



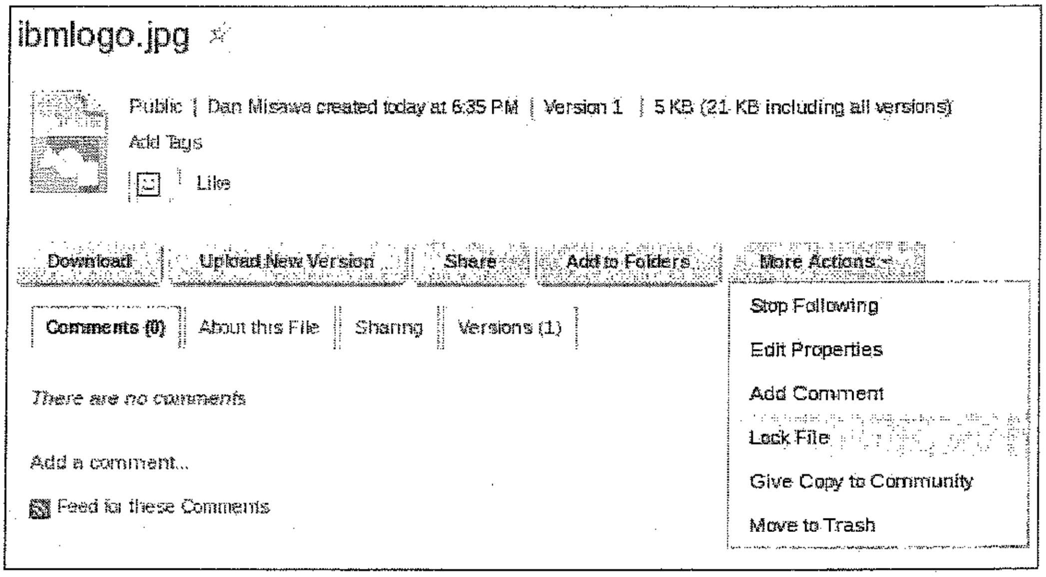
Step 16 Click the **Share** button. This will launch a **Share File** window. Using this window you can share a file with other users and/or communities. Note you have the option to specify if these people will be readers or owners of the file as you add them. Sharing in this way will send a notification into the activity stream of the selected users. For now you can click the **Cancel** button to close this window.



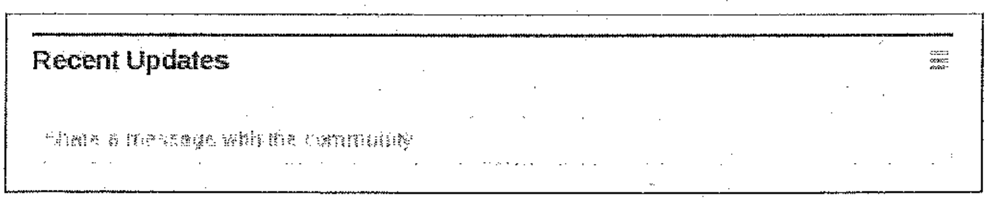
Step 17 Back at the File page click the **Add to Folders** button. This will open the **Add to Folders** window. Here you can add a copy of the specified file to any folder you have access to, including folders that are currently shared with you. (Depending on the user, there may be no folders listed.) Click the **Cancel** button to close the window.

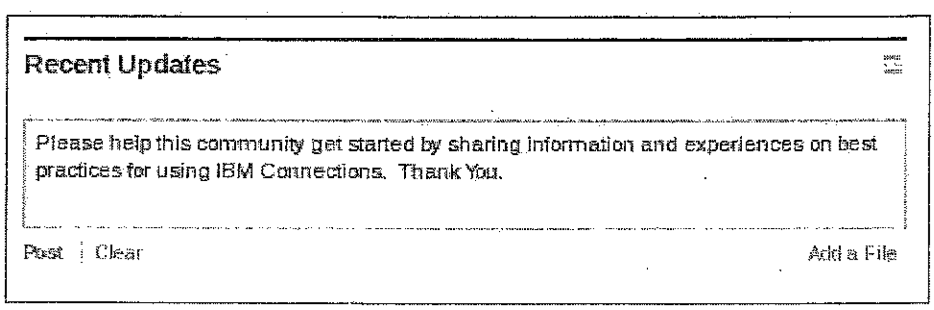


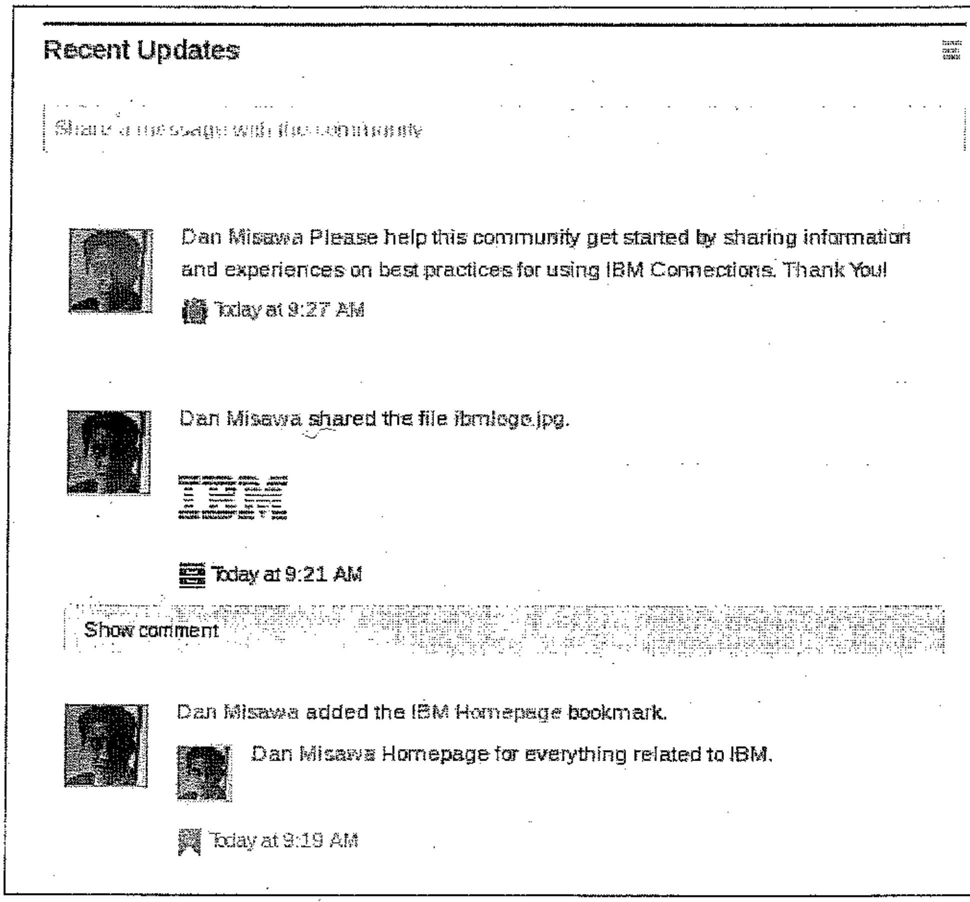
Step 18 Click the More Actions button. You should see a drop down menu with multiple options. The **Stop Following** option will keep future activity notifications from showing up in your activity streams. The **Edit Properties** option opens the **Edit Properties** window where you can modify the file name, extension, and description. The **Add Comments** option opens a comment text box for you to provide additional comments about the file that other users will see. The **Lock File** option is a new feature in Connections. This option allows the file owner to lock a file so no other users associated with the file can edit. Click the **Lock File** option and you will see a status message appear verifying the file is now locked. When a file is locked other users associated with the file will see that the file is locked and who locked the file. After locking the file once again click the **More Actions** button and now select **Unlock File**. This will once again unlock the file to allow edits by all associated owners. You should also see a status message letting you know the file is now unlocked. Clicking the **Give a Copy to Community** option will allow you to create a copy of the file and add it to a specific community. At that time you can edit the file name and any tags associated with the file. Finally , if you select the **Move to Trash** option this will move the specified file to the Trash section of the Connections Files application. Moving a file to trash makes the file unavailable to any user who might be associated. The file will only be moved to the trash area, but not yet deleted. In order to completely delete the file the user must navigate to the trash section of the Files area and confirm the deletion of this file.



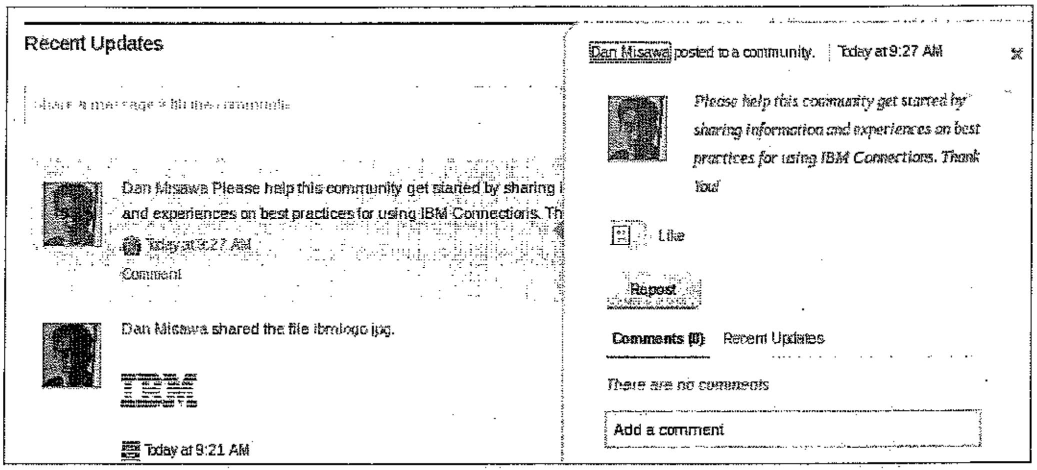
Step 19 The community page left navigation has many useful options to share information with the community or navigate through the community. Navigate back to the community homepage. Click on the **Recent Updates** option in the left navigation menu and you will see an activity stream of all recent activity happening within the community. For instance, Dan Misawa would like to encourage others to use the community so he makes a post to the community. Enter the message and then click **Post**.



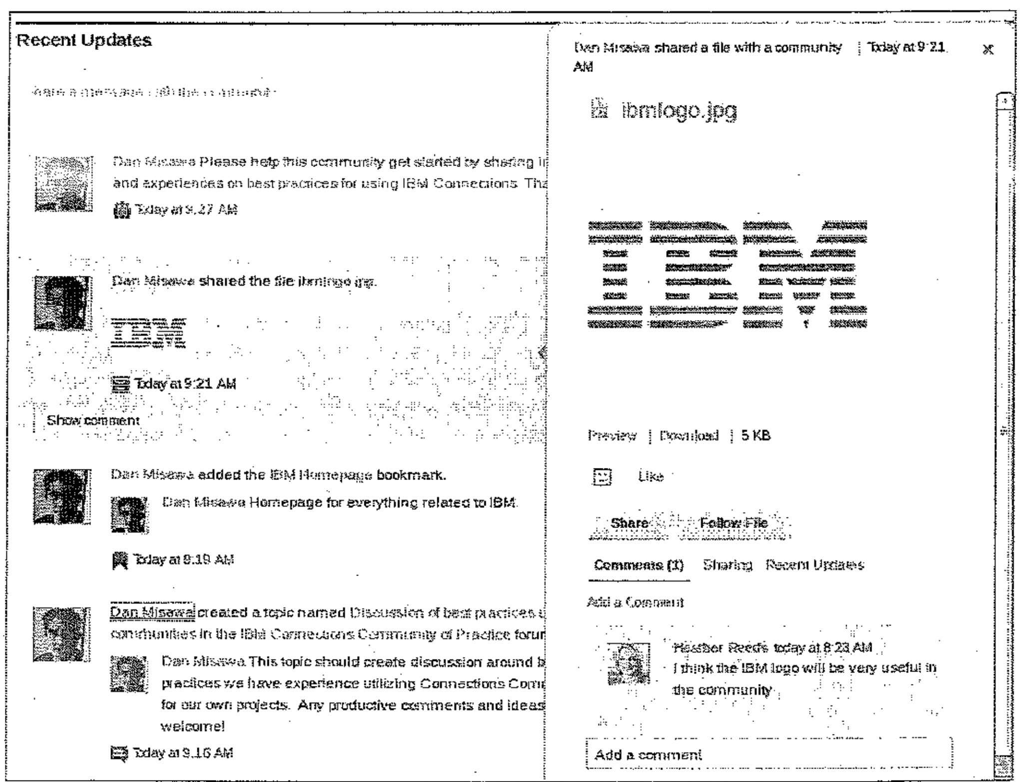




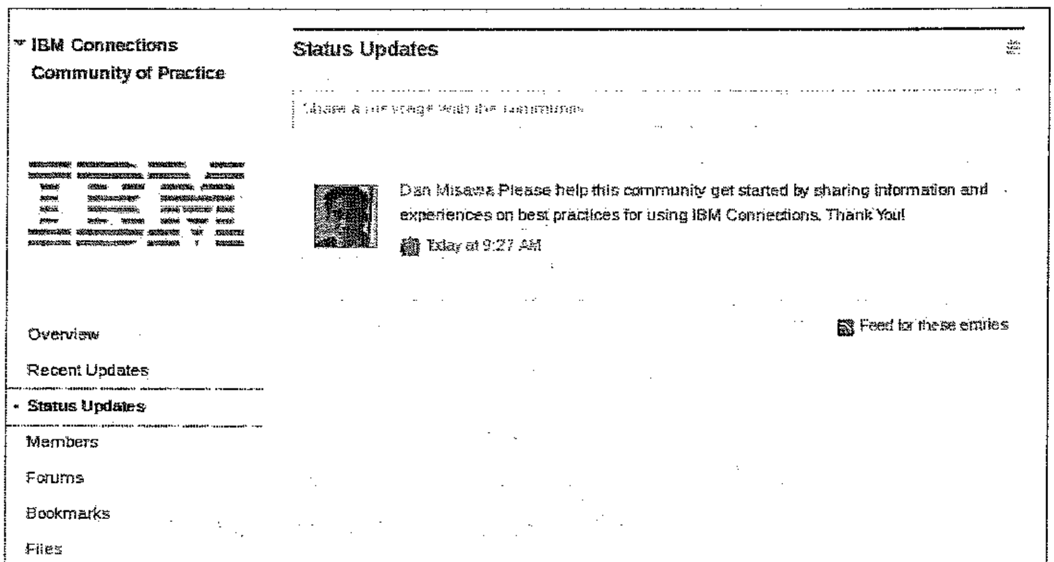
Step 20 Ask your neighbor to navigate to the **IBM Community of Practice** Community (Your community). As a new user to this community you would like to join the community so she can post messages, etc.. In the upper right corner click on **Join this Community**. A confirmation message is display welcoming her to the community. Users can also interact with recent activity in various ways through the embedded experience window. Clicking on an update within the **Recent Updates** activity stream will bring up an embedded experience window which gives a user more options to provide feedback to other users and the community. For example, you can "Like" a post, re-post to your followers, and add comments to the selected activity.



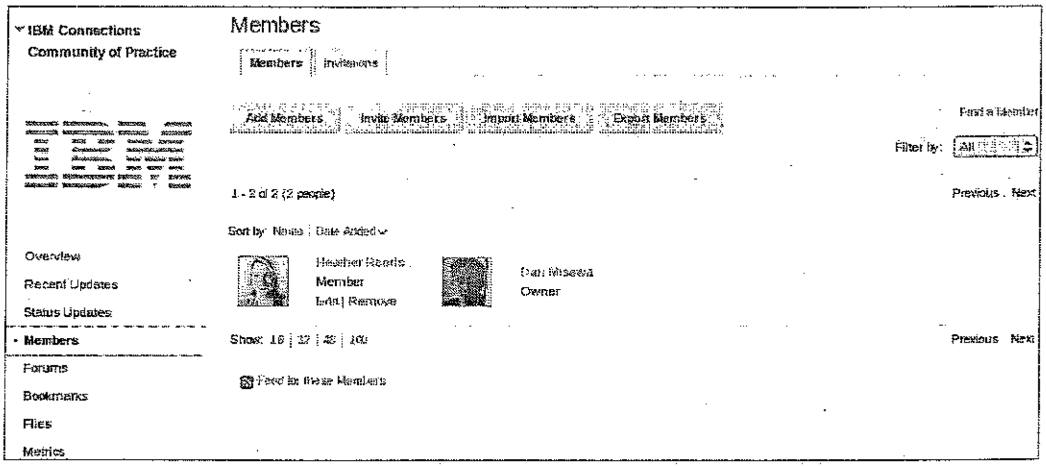
Step 21 In addition, you can interact with any posts that share files by clicking that update within the activity stream and launching the embedded experience window. Here you will have various options to interact such as ability to "**Like**" the files, **Share** the file for your other followers, **Follow File** to keep up with changes in your activity stream, **Comment** on the update, and also **Preview** or **Download** the file. Try using a few of these options to interact with a post.



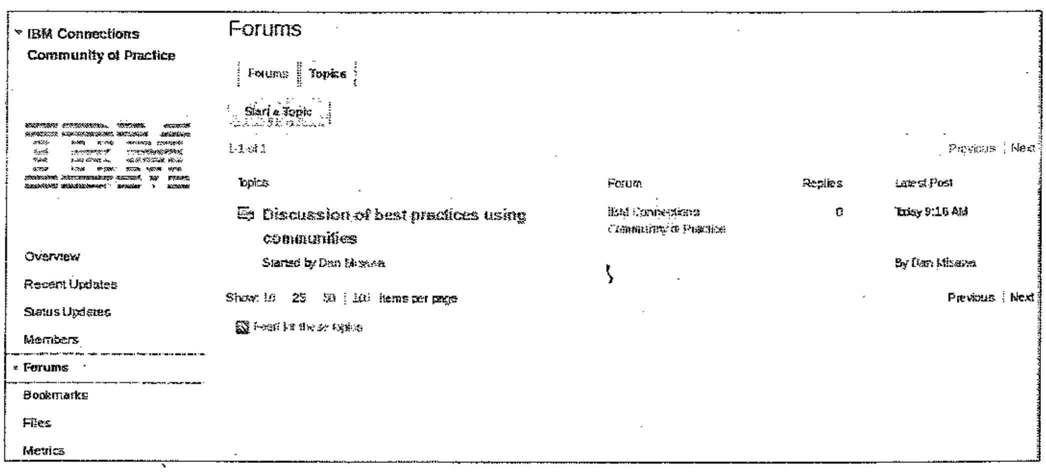
Step 22 Click the **Status Updates** section in the left navigation community menu. This view will filter out other types of community interaction from the activity stream and show you only posts that are classified as status updates. (There may be no status updates. This is dependent on the activity done with the image.) As with any activity stream interaction, in this view you also have the option to select an activity posted and bring up the embedded experience window which shows additional options to interact with a specific post.



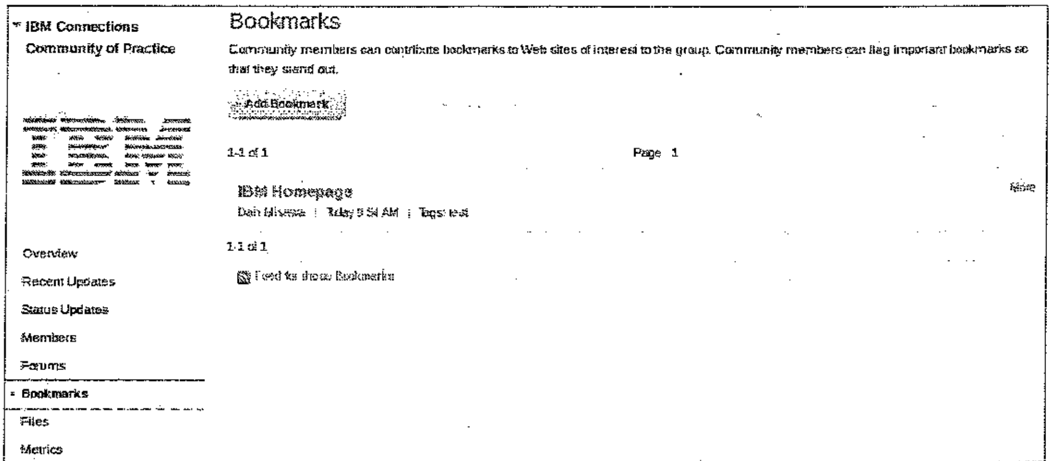
Step 23 Navigate to the **IBM Connections Community of Practice** community. Click on the **Members** menu selection in the left side community navigation menu. This will show you a view of all the current community members and their role in the community. This section is the primary area for community member management. Here you can **Add Members, Invite Members, Import Members** list, and **Export Members** list.



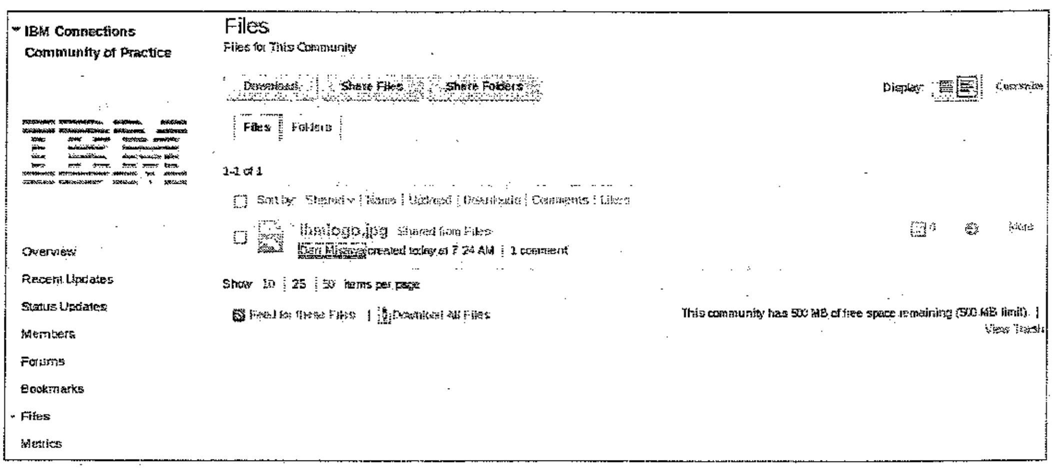
Step 24 Click on the **Forums** option in the left navigation menu. This will take you to the Forums area of the community. This is the same as clicking the View All link in the Forums section of the community homepage. This exclusive Forums navigation view lets you interact with community Forums and Topics with all the editor options previously mentioned. For example, you can navigate to a topic and edit what has been posted, or even lock the topic so close it from new posts.



Step 25 Click on the **Bookmarks** option in the left navigation menu. This will bring you to the Bookmarks section of the community. Similarly you can click the **View All** link from the **Bookmarks** section of the community homepage to reach this area. This area lets you add bookmarks to the community, or edit existing community bookmarks.



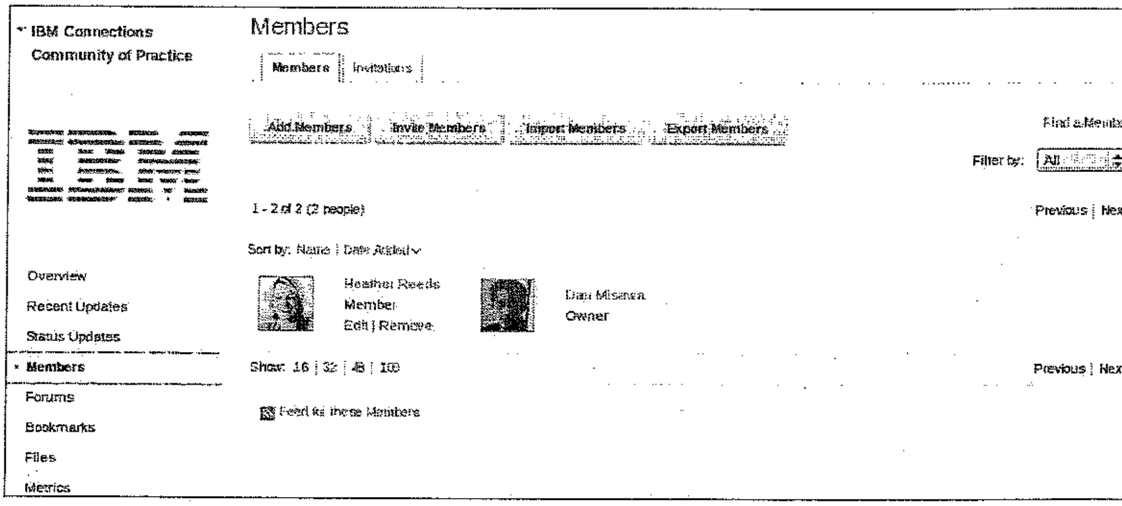
Step 26 Click on the **Files** option in the left navigation menu. This will bring you to the **Files** section of the community. Alternatively you can click the **View All** link from the **Files** section of the community homepage to reach this area. Here you can manage current community shared files, download files, and share files and folders with the community.



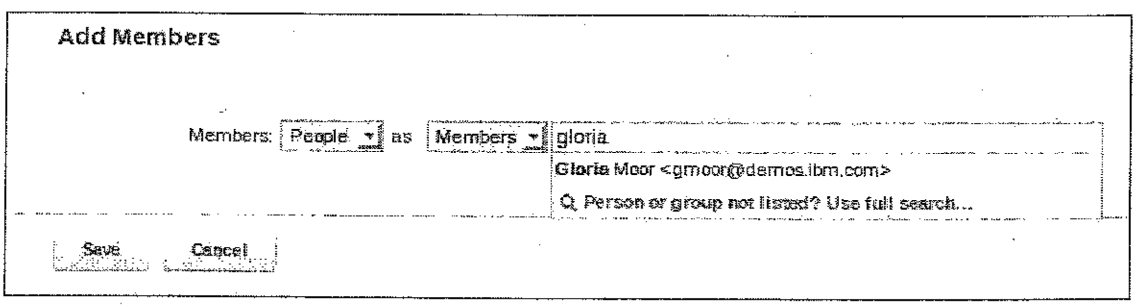
Community User Management

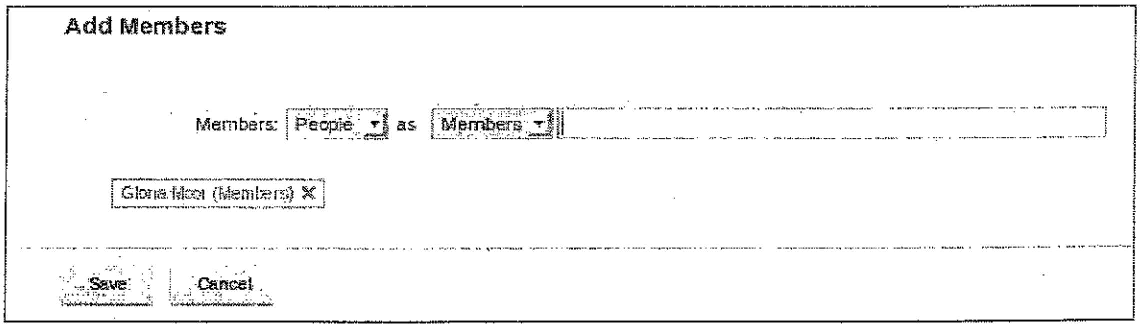
In this section we will explore the activity of managing users as the community owner.

Step 1 Navigate to the **Members** section of your community.

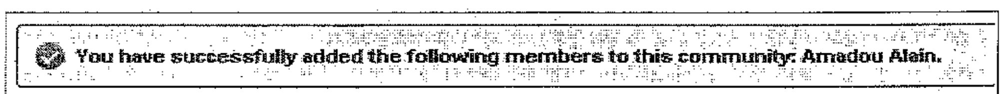


Step 2 Click on the **Add Members** button to add a user to the community. On the Add Members screen you have the option to add members to the community as individual users or a group of users. You also have the option to add these people as Members or Owners. Add a single user to the community as a member. You can use the text box on the right to type the person's name (e.g. **gloria**) and notice how the type ahead brings up relevant user names. Select the pop up name and it will be select to be added. Click the **Save** button.

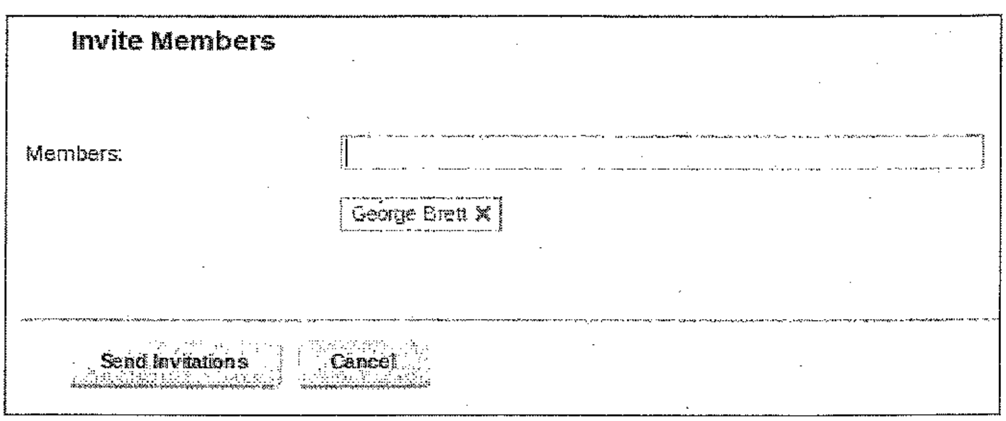




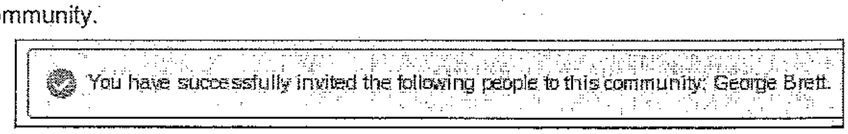
Step 3 You will now see a confirmation message letting you know the user has been added to the community.



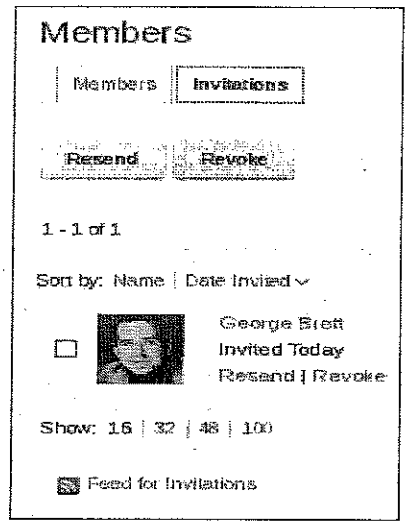
Step 4 Click the **Invite Members** button. You will see the Invite Members page open with a text box allowing you to type in a member's name to invite to this community. This text box also provides type ahead. Type in a users name (e.g. **George Brett**) and click the **Send Invitations** button.

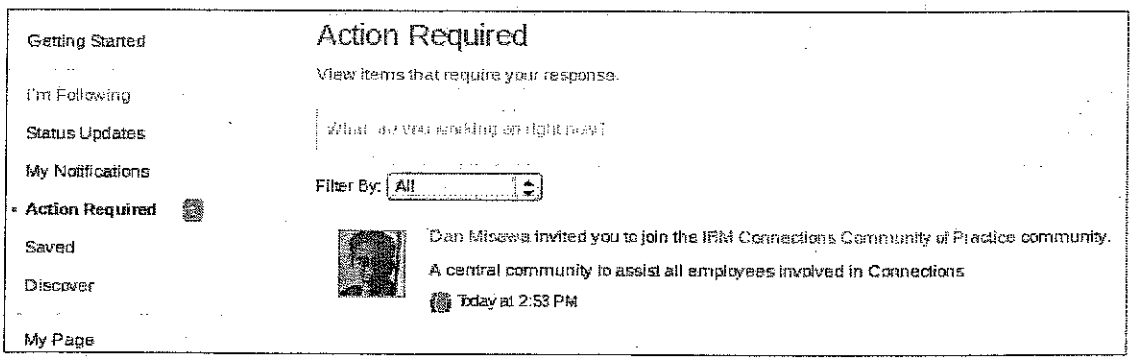


Step 5 You will now see a confirmation message letting you know the user has been invited to the community.



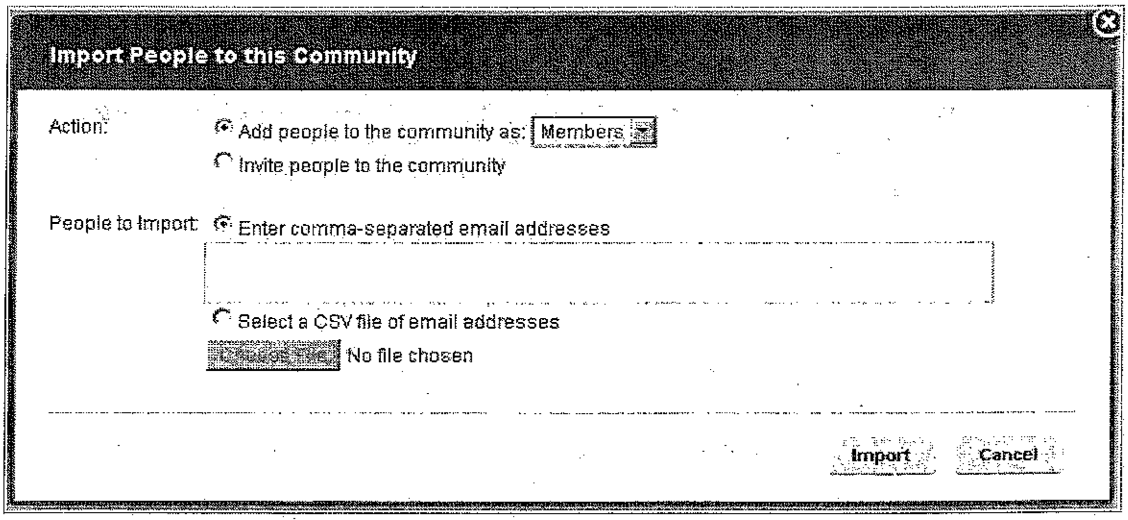
Step 6 If you click the Invitations tab in the main Members page you will see a list of all pending invitations. You also have the options to resend or revoke a community invitation to a user by clicking the Resend and Revoke tabs respectively.





Step 7 Once a user is sent an invitation an Action Required notification will be shown on their activity stream. From the **Home** Connections screen, clicking on the **Action Required** view will allow the user to see the invitation. Hovering your mouse over the invitation will all you to accept or reject the community invitation. Go ahead and click **Join this Community**.

Step 8 Navigate to the **IBM Connections Community of Practice** community. From the Members section of the community you can add large groups of users by importing a member list. Click the Import Members button. You will be presented with the Import People to the Community window where you have the option providing several user email addresses or upload a CSV file of email addresses. This method allows you to quickly add a large number of people to a community. This option also allows you to add these users as either Members or Owners of the community with a drop-down box in the Action section of the window.

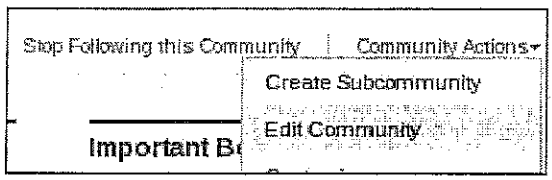


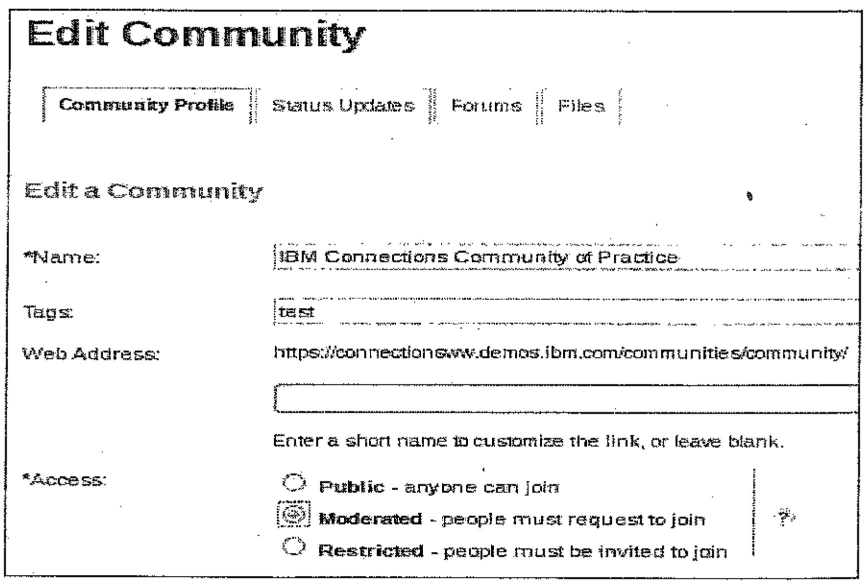
Step 9 From the Members section of the community you can also chose to export your existing owners and members. Click the **Export** Members button. You will see the Export people from this Community to a CSV file window. You can chose to export all user records associated with this community, or specify only owners, or only members, by using the corresponding check box. Once you have made your selection you can click the **Export** button to create the CSV file and save to your local computer. This action will create a membership.csv file by default in your downloads directory. Press **Cancel** and go back to Community home page

Joining a Community

Joining a community is an easy way to get more involved with activities and other users with a common goal or interest. In this section we show the procedure of becoming a part of a Connections community.

Step 1 Update the community access for the IBM Connections Community of Practice community from the user that originally created this community (e.g. Dan Misawa). Go back to the Community Homepage. In the upper right corner, select **Community Actions** then **Edit Community.** Click the radio button for **Moderated**, then click **Save.** This also shows you that you can change the access from public to moderated after a community was created.

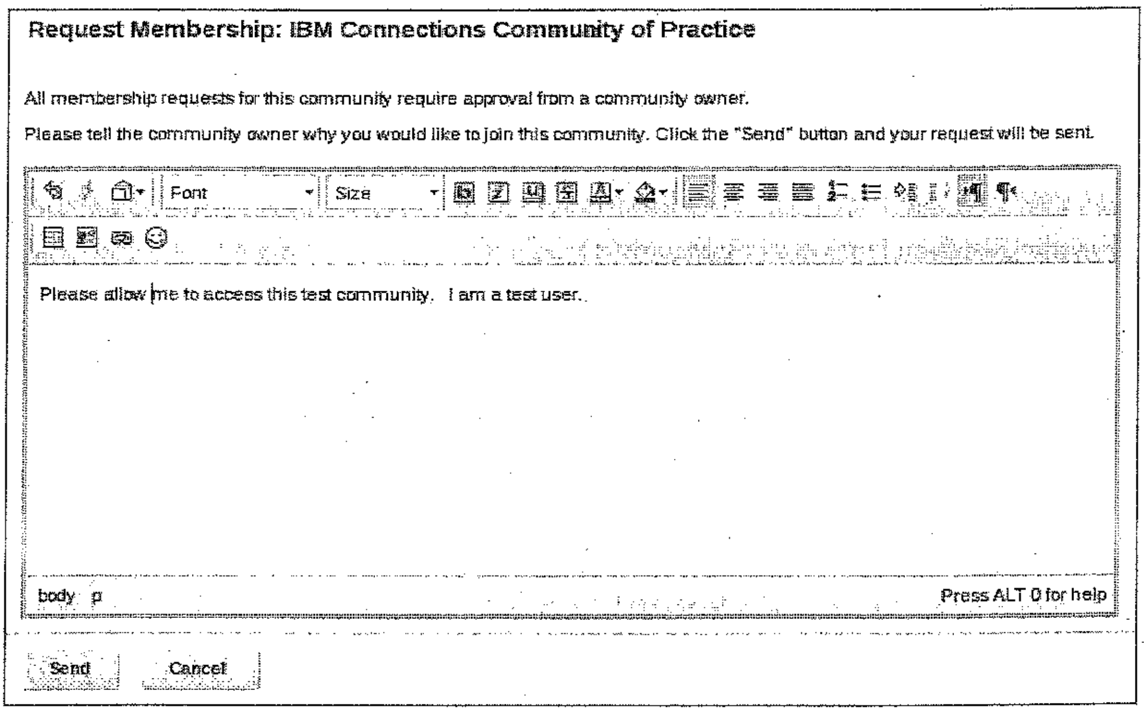


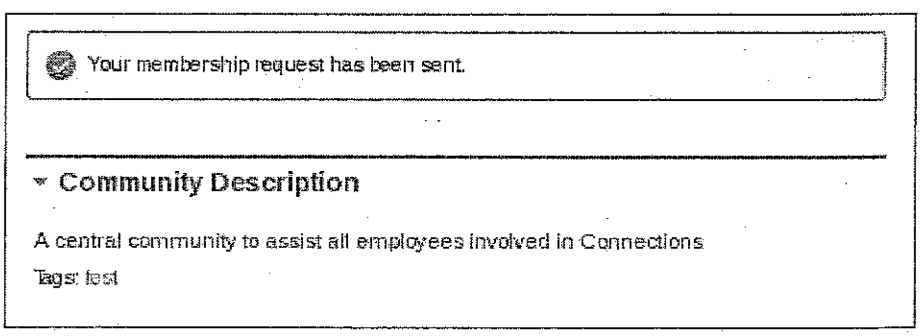


Step 2 Ask your neighbor to navigate to public Communites and you'll see the **IBM Connections Community of Practice** in the list.. As a suggestion, try using the search feature to find the community, if you tagged it with **test**. Click on the Community. Notice that the community is now a Moderated community by the icon next to the community name.

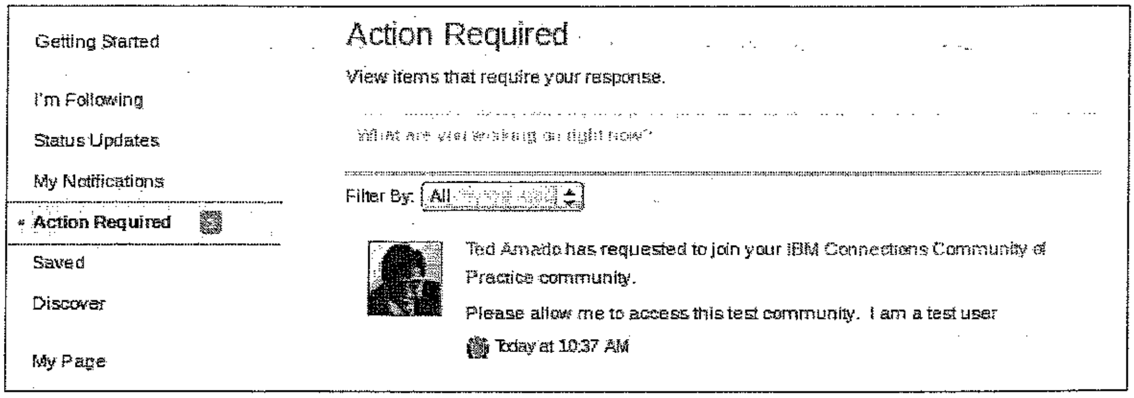
Step 3 Click the **Request to Join this Community** button located in the top right area of the community homepage.

Step 4 On the Request Membership page enter your justification and click the **Send** button





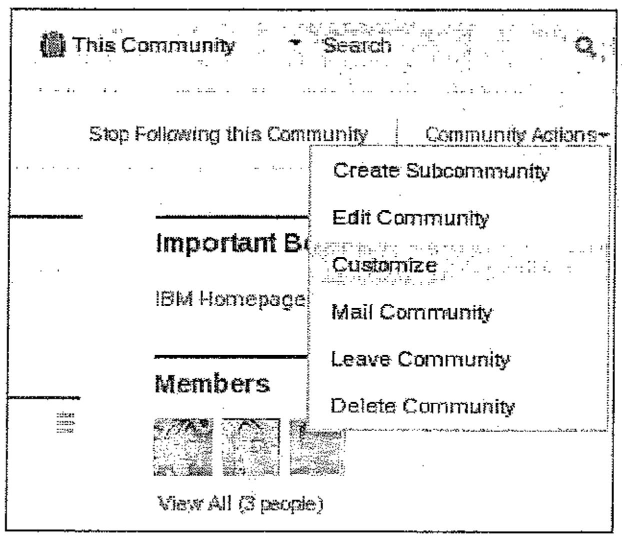
Step 5 You will see a confirmation message that your membership request was sent to the community owners. From the Connections **Home**, click on **Action Required**. and you will see the request to accept the membership within the the Activity Stream.


Getting Started
I'm Following
Status Updates
My Notifications
Action Required
Saved
Discover
My Page
Action Required
View items that require your response.
What are you working on right now?
Filter By: All
Ted Amado has requested to join your IBM Connections Community of Practice community.
Please allow me to access this test community. I am a test user
Today at 10:37 AM

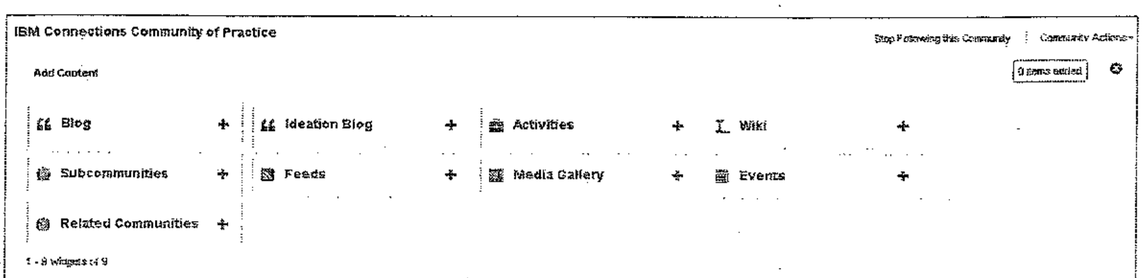
EVENTS IN A CONNECTIONS COMMUNITY

Events is a new feature of IBM Connections. In this section we walk through how to include the Events section in your community, how to create an event, and event interaction.

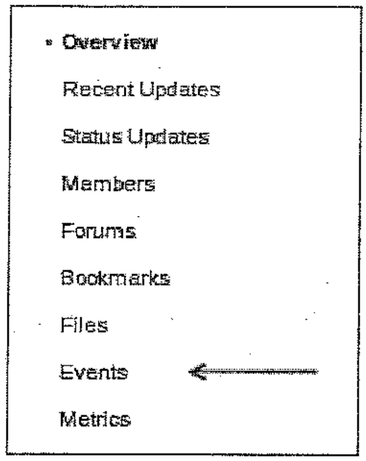
Step 1 Within the **IBM Connections Community of Practice** community, click the **Community Actions** drop down menu located in the top right section of the browser window. Select **Customize**.



Step 2 You will see the Add Content window at the top of your community page with a number of possible widgets you can add to your current community. Click the plus sign next to the Events selection.

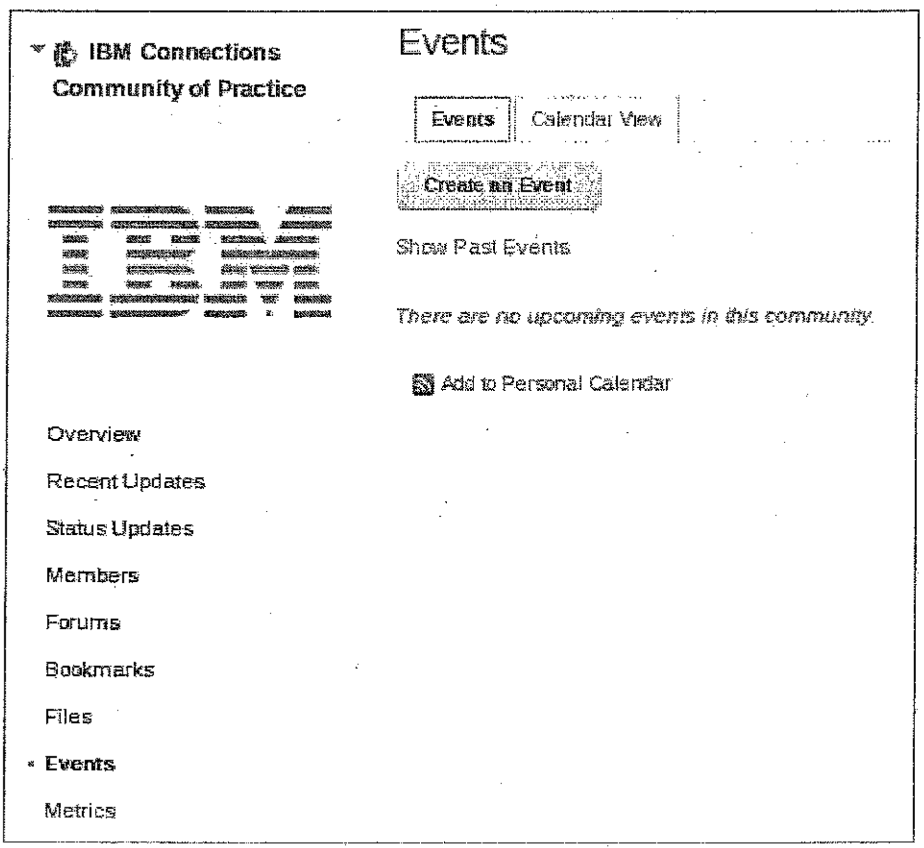


Step 3 After selecting Events you will briefly see a status telling you the widget is being added to your community. Once this is done the Events option will change to show **Events** added. You will also see a note stating 1 items added in the upper right corner of the window. Most importantly, the Events section of the community has now been added to the left navigation menu of the community site.



Step 4 You can now close the Add Content window by clicking the X in the upper right corner.

Step 5 In the main navigation menu on the left of the community page click the newly created **Events** option. This will take you to the Events page of this community.

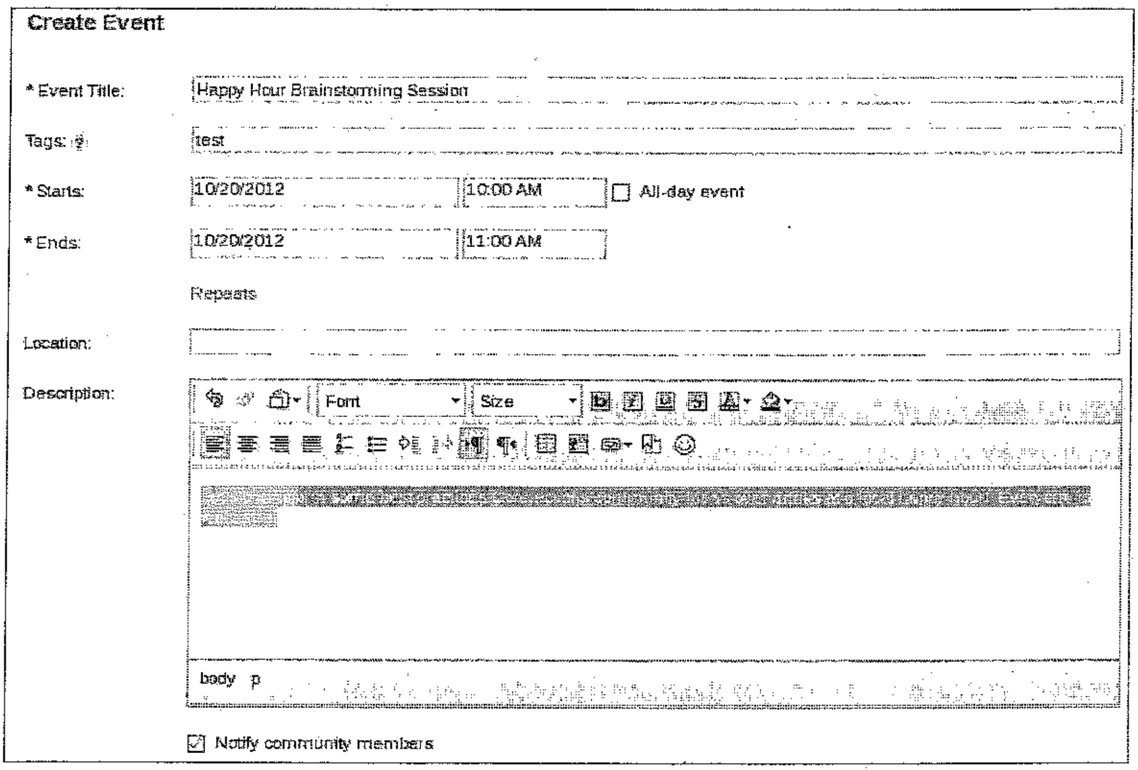


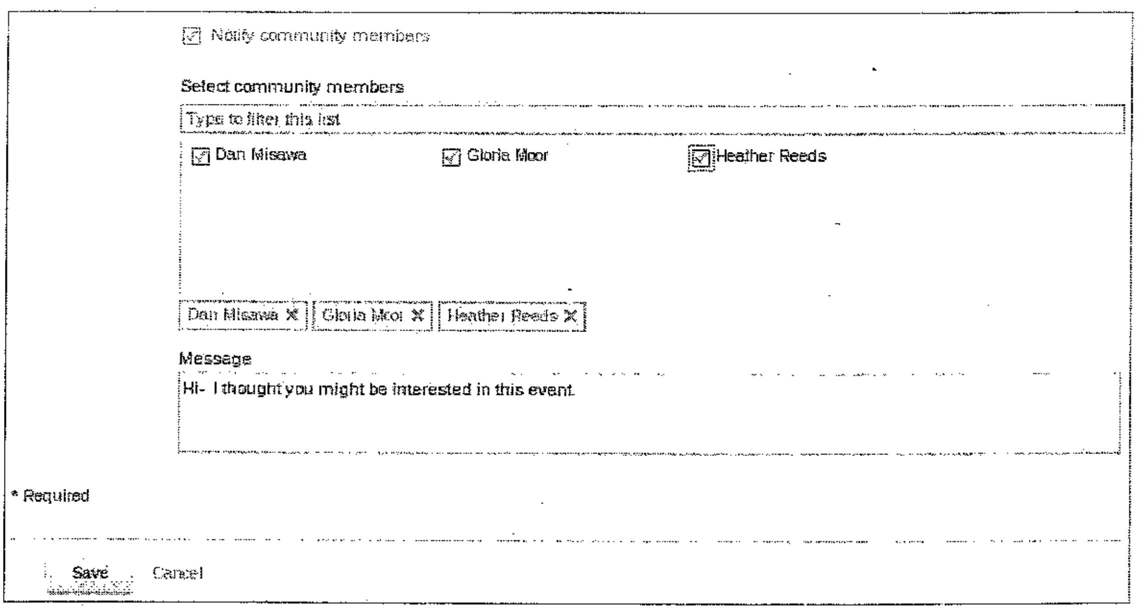
Step 6 Get started by creating an event! Click the **Create an Event** button. You are presented with the Create Event page. Provide all necessary information for your event and provide a description and possible location.

Step 7 Be sure to indicate the event for a future date..

Step 8 Note an important option at the bottom of this page. You have a check box for Notify community members. Put a check mark in this box to expand the page and see a list of users with check mark options beside their names to flag for notification of this new event. The users selected will get get notifications of this event complete with a custom message you provide in the Message text field of the page. **Be sure to select Your neighbor, e.g. Heather Reeds.**

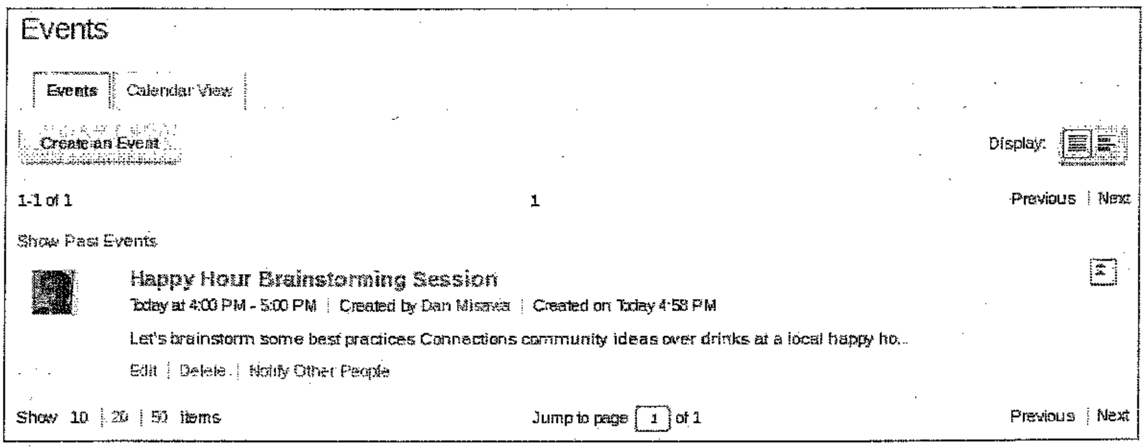
Step 9 Once you have completed the options on this events creation page click the **Save** button.



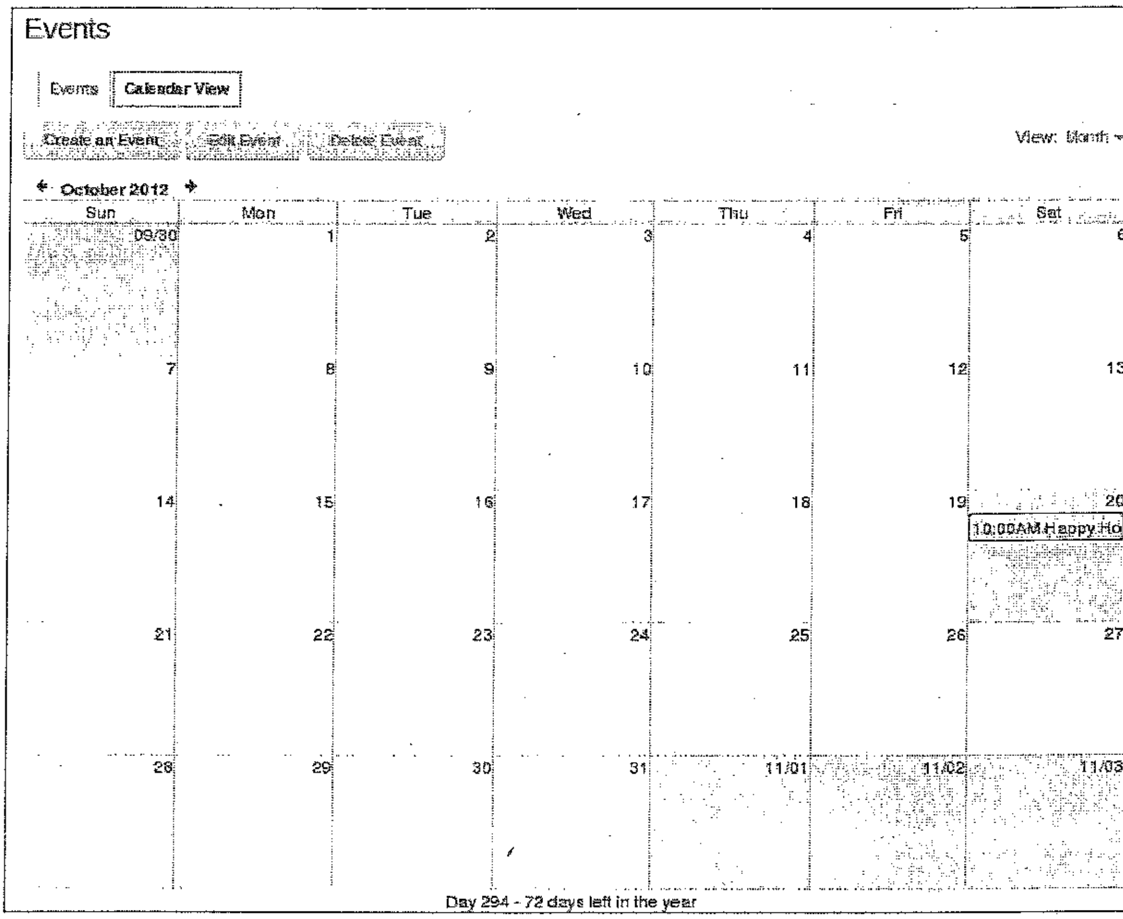


Step 10 You will be taken back to the main Events community page where you will see the newly created event. (Hint, If the event is not displayed, click on the option to switch between display

past events and upcoming.) Note to the right of the event you will see arrows pointing down. Click on the arrows to expand the details of this event. You will also see links for editing this event which include Edit, Delete, and Notify Other people.



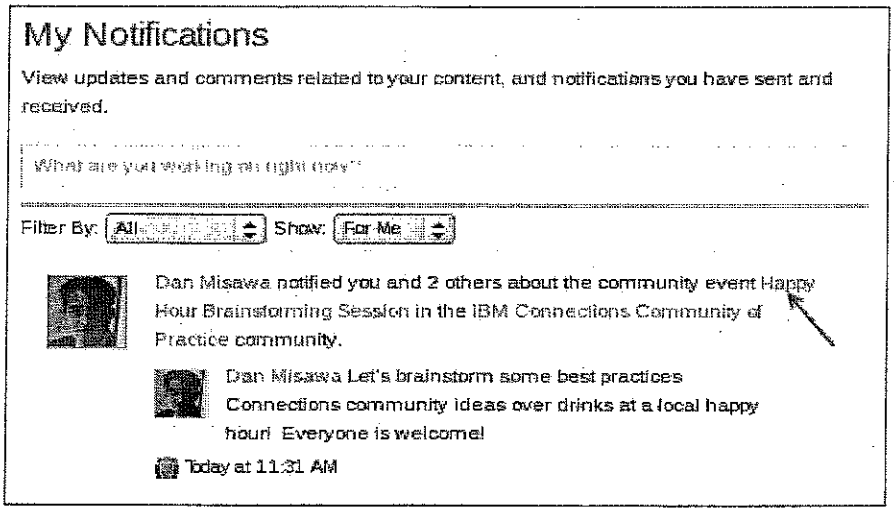
Step 11 From the main Events page click on the **Calendar View** tab, located at the top of the window. The Calendar View will show you a customizable view of events. By clicking on the View drop down menu located to the right you can switch to a view of One Day, Two Days, Five Days, Week, and Month.

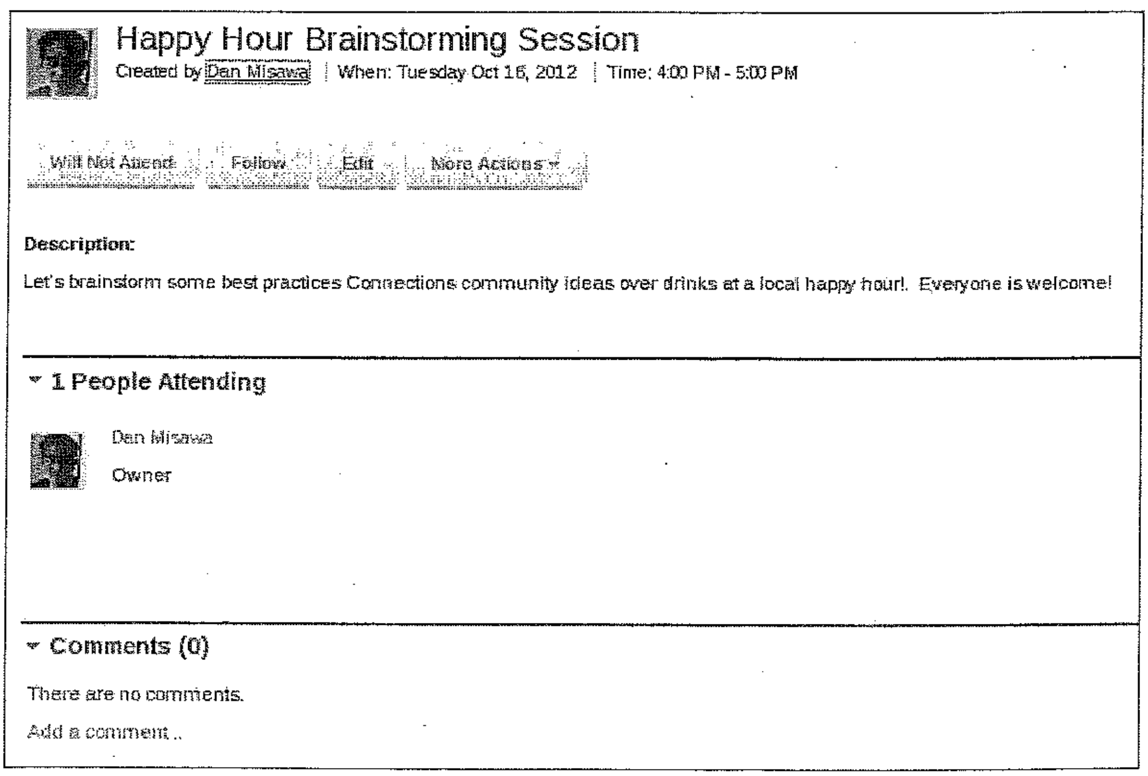


Step 12 In the Calendar View you also have the option to create a community event by clicking the **Create** and **Event** button, located at the top of the window. By clicking on an event in the calendar you will highlight the **Edit Event** and **Delete Event** buttons. These buttons will allow you to alter a scheduled event right from the calendar view as needed. Under these buttons you should note the label for the current month being displayed, and arrows on either side of this label. Using all three of these controls will allow you to move forward or backward by month, or navigate directly to a specific month, week, or day.

Step 13 The member (e.g. Dan Misawa) that created the event in the community will be notified of the new event through his Connections activity stream. (Hint, click on **Recent Updates** within the community to see the event.) Invited members will also receive an email communication with the same information.

Step 14 From Connections **Home** click on **My Notifications**. A community member can click on the event link to take them directly to the community Event page. Here they can provide feedback that they are attending the event by clicking the **Will Attend** button. Members also have the option to follow changes to the event in their activity stream by clicking the Follow button. Using the **More Actions** button also provides the option to Notify Other People of this event, or even Delete or Edit the event if the user has proper authority.

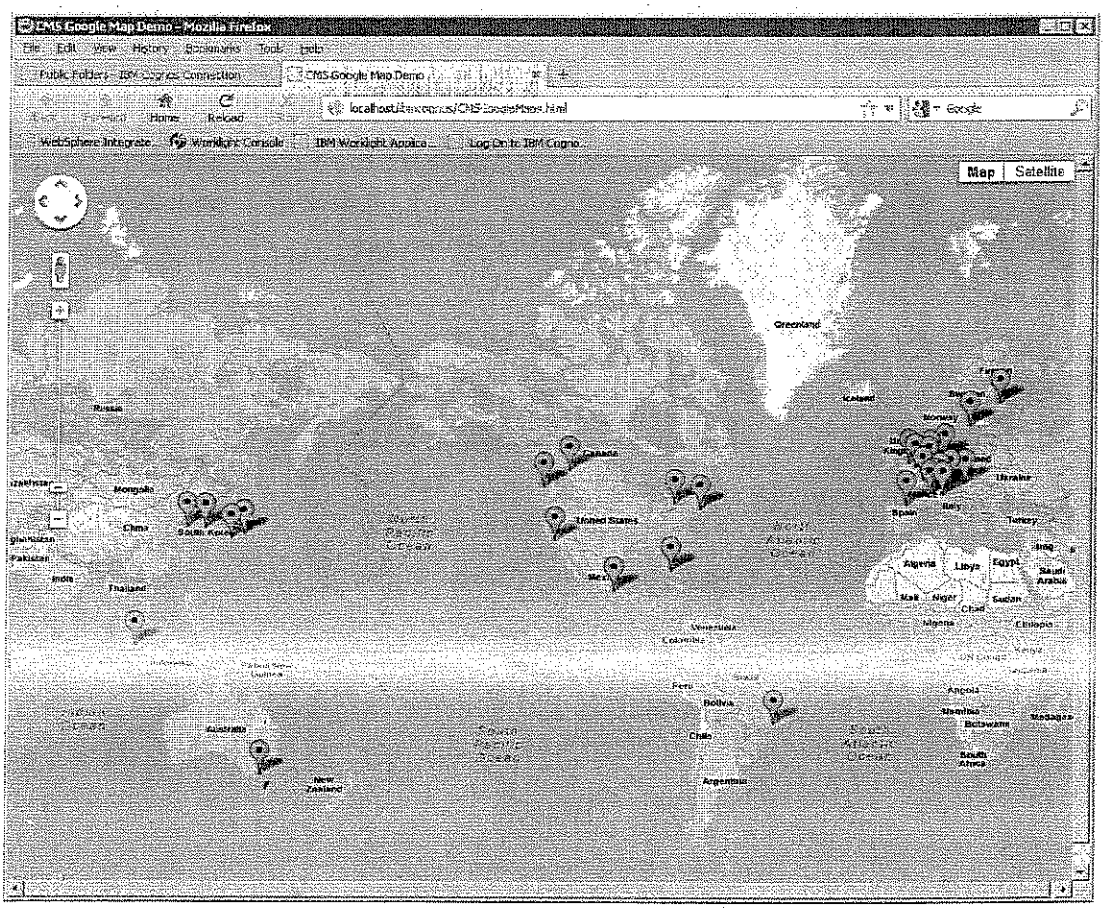




This concludes the Connections 4 Features Walk Through lab. Additional information will be added to this lab to explore more features of Connections. For more information about Connections 4, be sure to check the Connections 4 product wiki

Ch2 BA Lab Material

In this exercise you will be leveraging the Cognos Mashup Service to access data from the Cognos platform on the Cloud and integrate with the Google Map service on the HTML pages located in your machine.

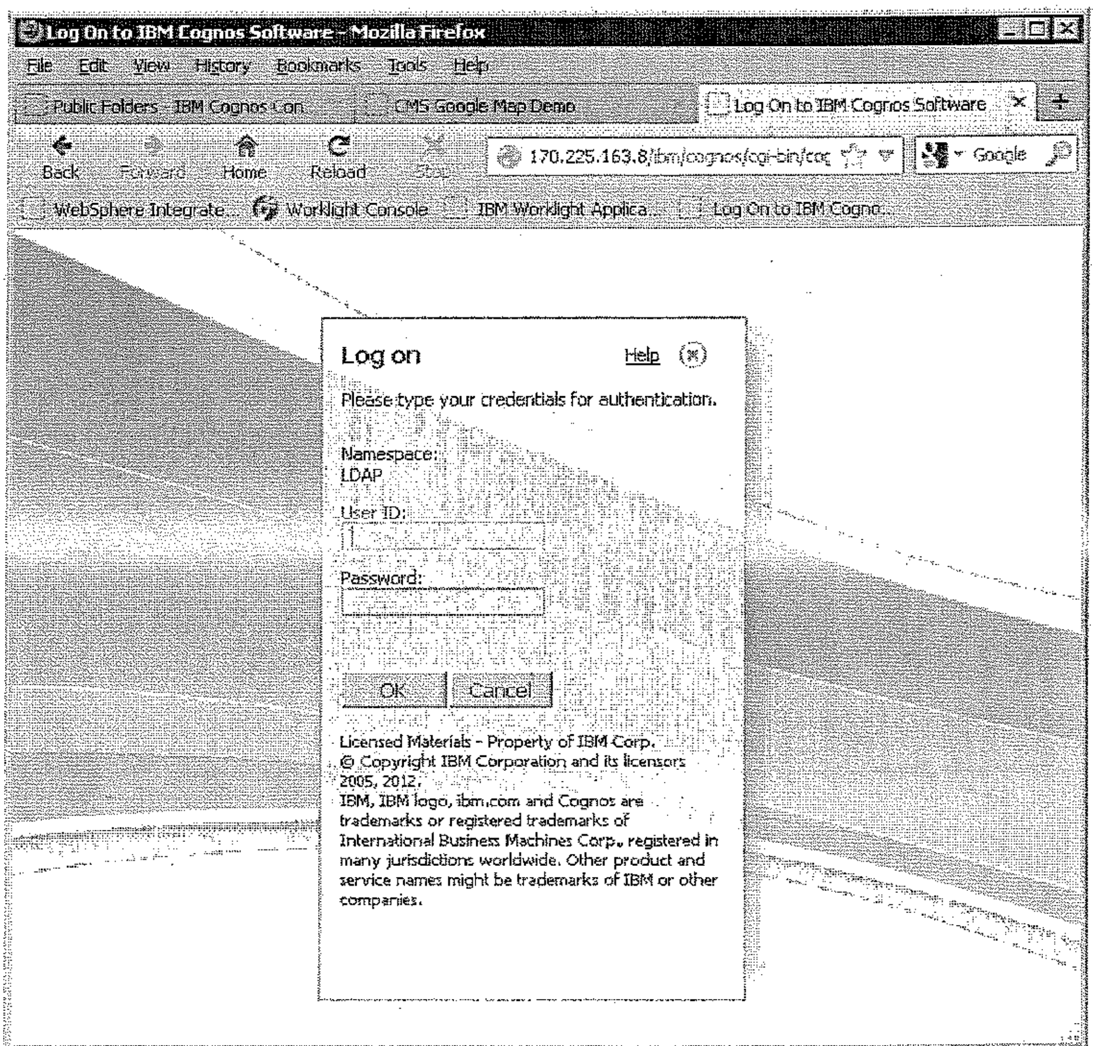


Before the exercise, you should examine the reports on the BI Cloud server

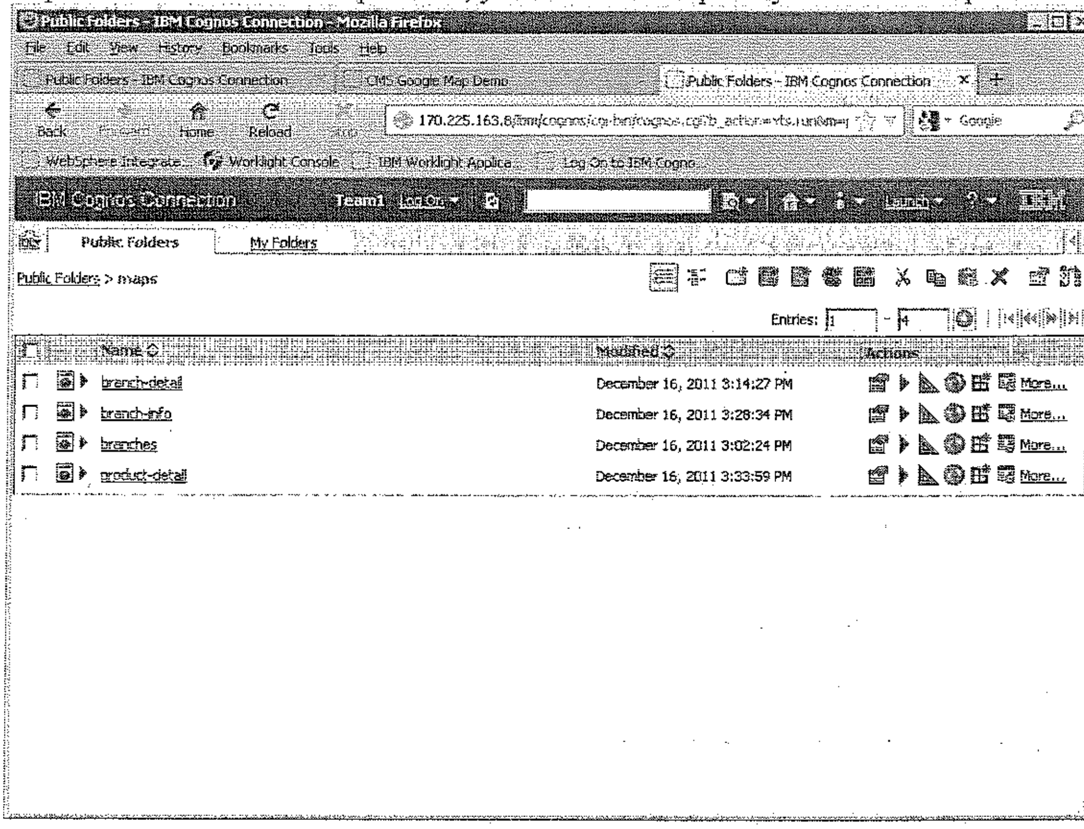
Open the following URL in Firefox
http://(IP to be provide)/ibm/cognos/cgi-bin/cognos.cgi

Login using the following credential.

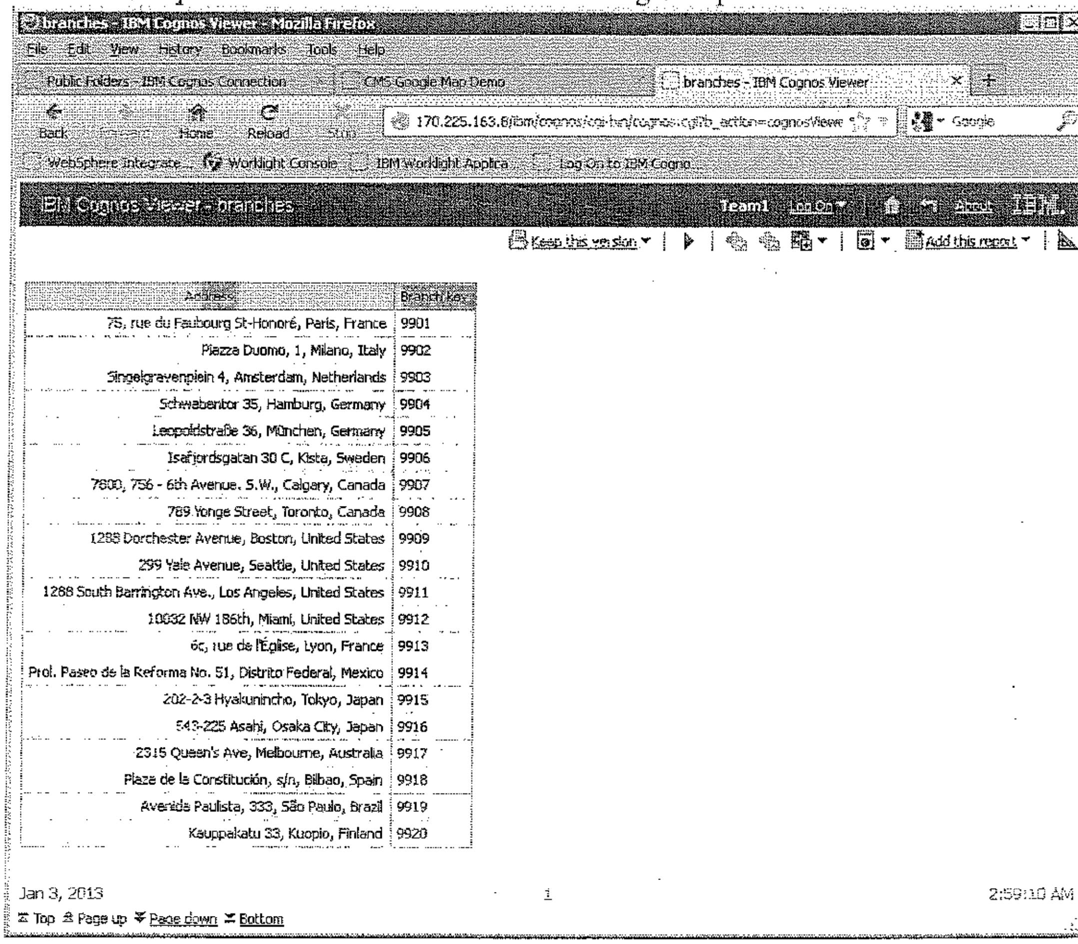
Namespace: OPENLDAP
Login name: team1
Password: password



Report located inside the maps folder, you can run the report by click on the report.



“branches” report is used to define locations to Google map

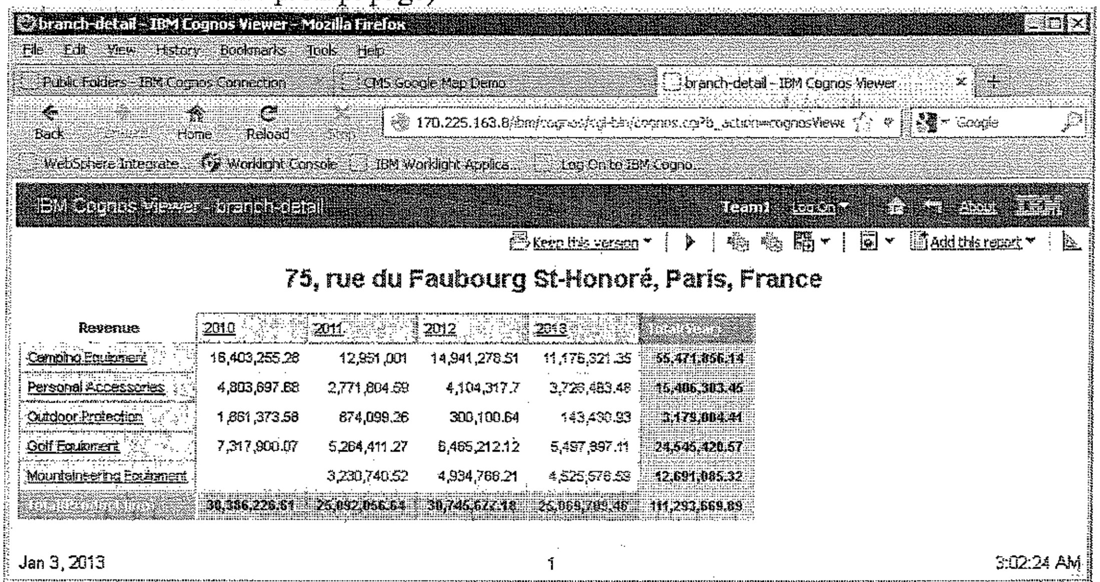


Address	Branch key
75, rue du Faubourg St-Honoré, Paris, France	9901
Piazza Duomo, 1, Milano, Italy	9902
Singelgravenplein 4, Amsterdam, Netherlands	9903
Schwabentor 35, Hamburg, Germany	9904
Leopoldstraße 36, München, Germany	9905
Isafjordsgatan 30 C, Kista, Sweden	9906
7600, 756 - 6th Avenue. S.W., Calgary, Canada	9907
789 Yonge Street, Toronto, Canada	9908
1288 Dorchester Avenue, Boston, United States	9909
299 Yale Avenue, Seattle, United States	9910
1288 South Barrington Ave., Los Angeles, United States	9911
10032 NW 186th, Miami, United States	9912
6c, rue de l'Église, Lyon, France	9913
Prol. Paseo de la Reforma No. 51, Distrito Federal, Mexico	9914
202-2-3 Hyakunincho, Tokyo, Japan	9915
543-225 Asahi, Osaka City, Japan	9916
2315 Queen's Ave, Melbourne, Australia	9917
Plaza de la Constitución, s/n, Bilbao, Spain	9918
Avenida Paulista, 333, São Paulo, Brazil	9919
Kauppakatu 33, Kuopio, Finland	9920

Jan 3, 2013 1 2:59:10 AM

Top Page up Page down Bottom

"branch-detail" report is used to define product details of branch. (input "9901" as the branch number in the prompt page)

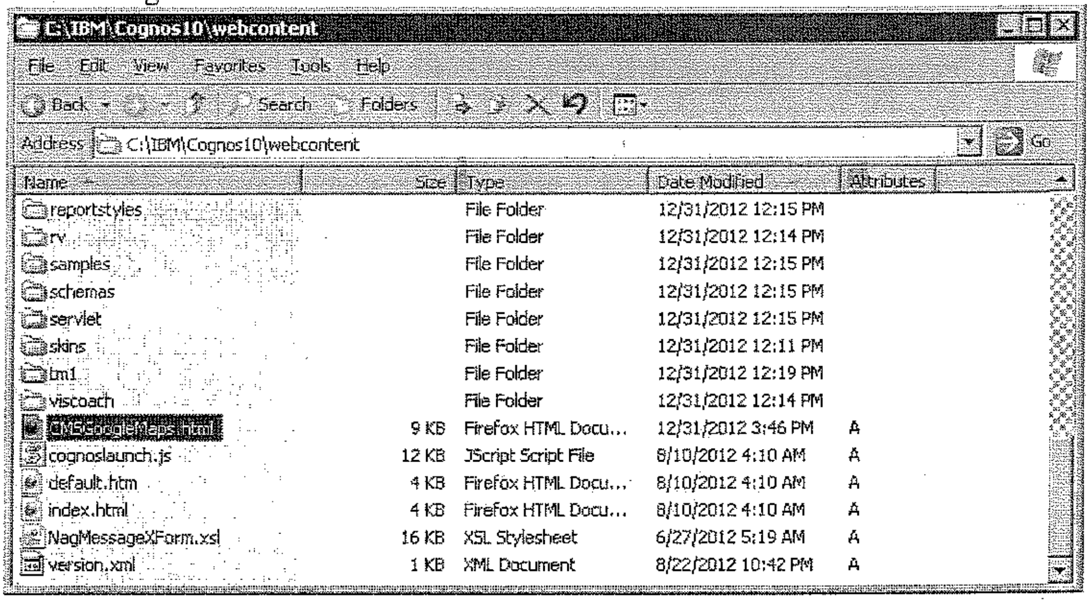


75, rue du Faubourg St-Honoré, Paris, France

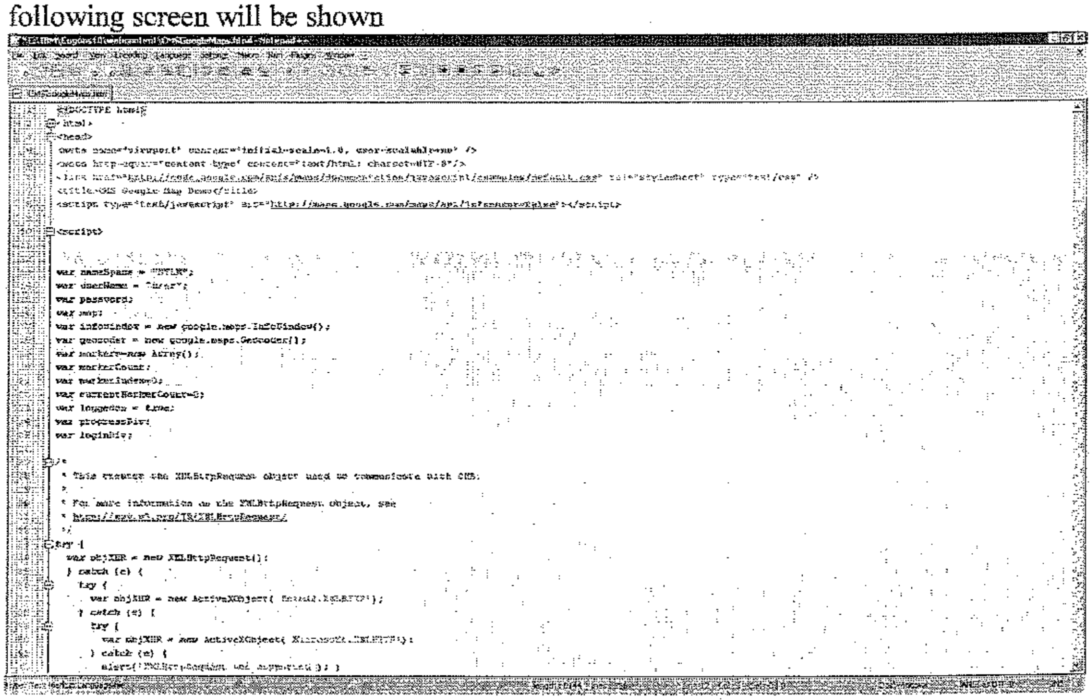
Revenue	2010	2011	2012	2013	[illegible]
Camping Equipment	16,403,255.28	12,951,001	14,941,278.51	11,176,321.35	55,471,856.14
Personal Accessories	4,803,697.68	2,771,804.59	4,104,317.7	3,726,483.48	15,406,303.45
Outdoor Protection	1,861,373.58	874,099.36	300,100.64	143,430.93	3,179,004.41
Golf Equipment	7,317,900.07	5,264,411.27	6,465,212.12	5,497,897.11	24,545,420.57
Mountaineering Equipment		3,230,740.52	4,934,768.21	4,525,576.59	12,691,085.32
[illegible]	30,386,226.61	25,092,056.64	30,745,677.18	25,069,789.46	111,293,669.89

Jan 3, 2013 1 3:02:24 AM

1. Copy CMSGoogleMaps.html you downloaded from Lotus connection to C:\IBM\Cognos10\webcontent



2. Right click the CMSGoogleMaps.html and click "Edit with Notepad++", then the following screen will be shown



3. In order to connect to the BI server, you need to define the gateway.

<u>How to find the gateway?</u>
The Gateway you used to access the server in your Firefox is the public gateway to BI server. (http://(IP to be provide)/ibm/cognos/cgi-bin/cognos.cgi)

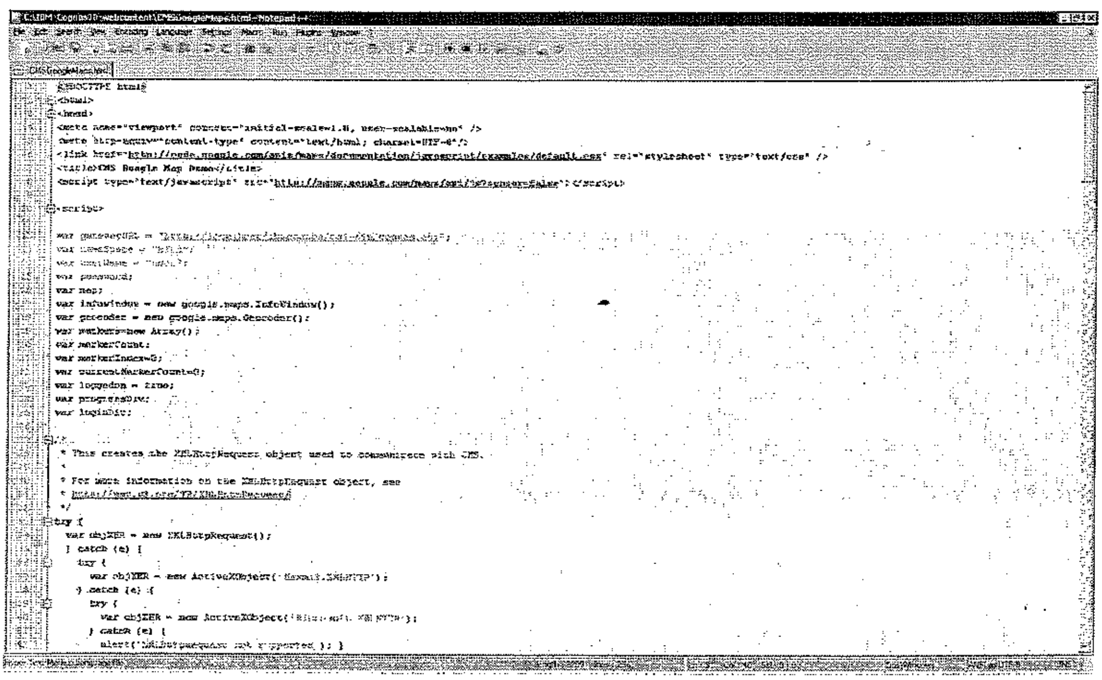
In order to allow the API to work, you need a local gateway.

Try open the following URL in your Firefox, it should return the same login pages as the public gateway

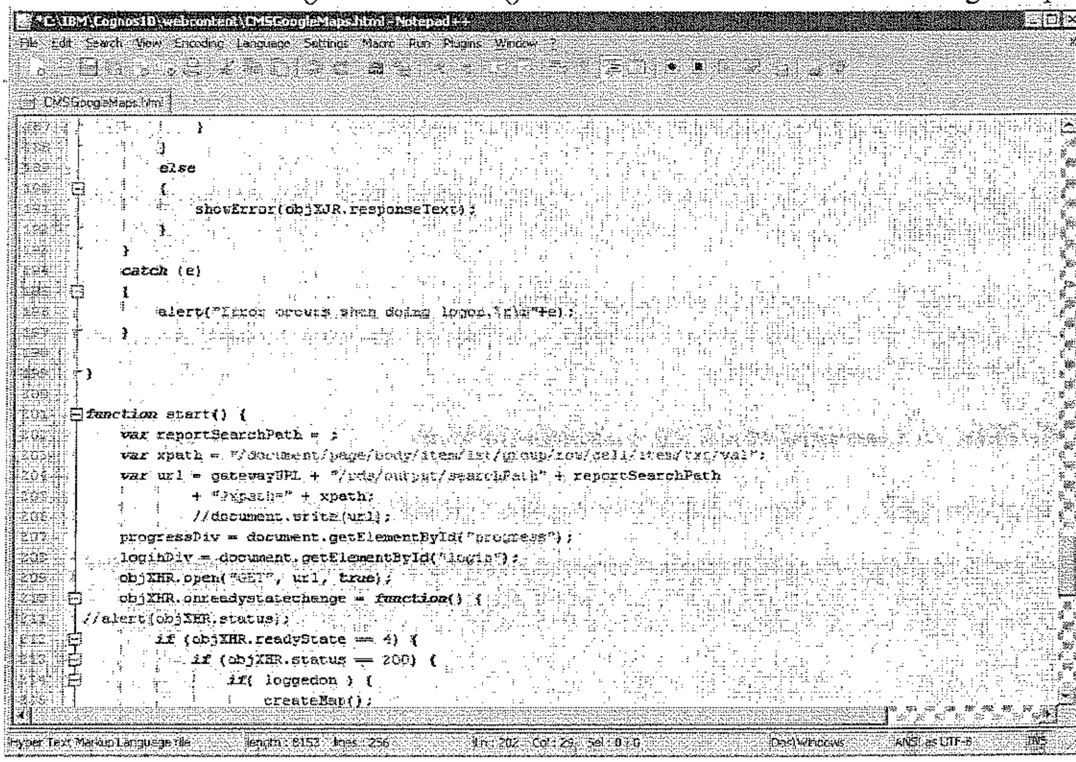
http://(local IP)/ibmcognos/cgi-bin/cognos.cgi
(For example the local IP can be 192.168.1.1, **IP has to be used rather than localhost for the HTML page to work)**

Therefore, define the gatewayURL variable as follow:

var gatewayURL = "http://(local IP)/ibmcognos/cgi-bin/cognos.cgi";

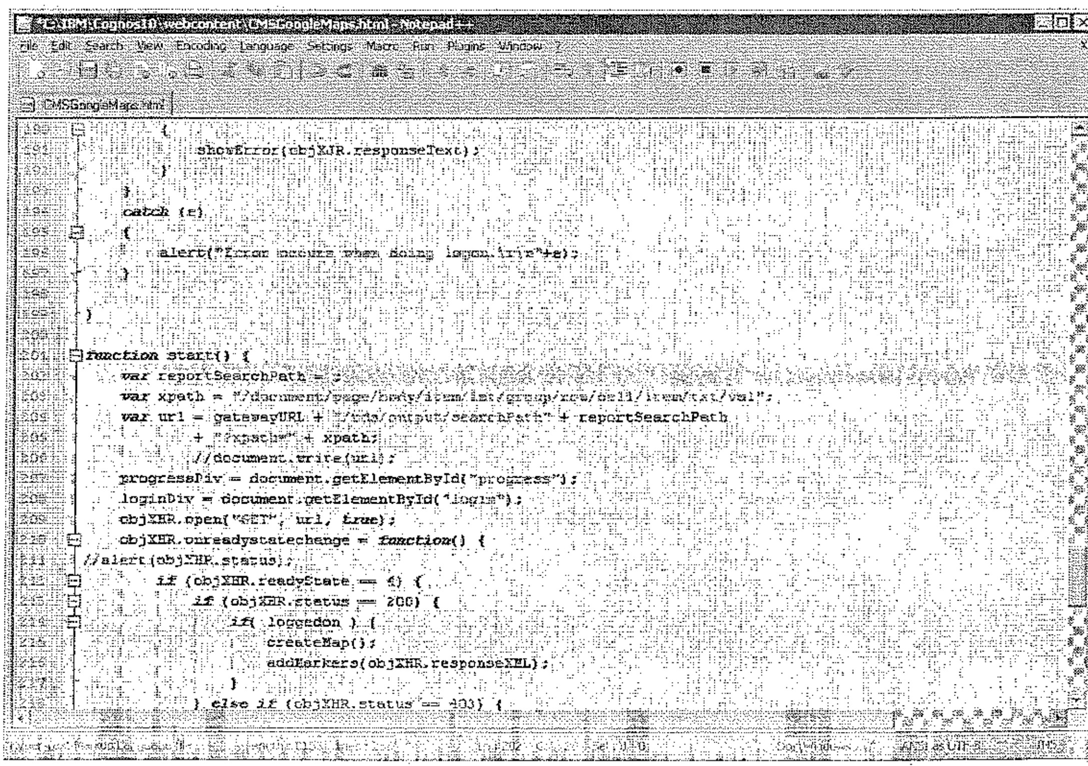


4.
Now you know the location of the report, you have to define report for the HTML pages. Search for function start(). Function start() is used to define the location of Google map.



Therefore, it will leverage "branch" report.
To define reportSearchPath variable, type as follow:

```
var reportSearchPath = "/content/folder[@name='maps']/report[@name='branches']";
```



Then search for function openInfo(), this function will leverage the "branch-detail" report. Please insert reportSearchPath variable with the same syntax as above.

```
    }

function getAddress(key) {
    for (i=0;i<markerCount;i++) {
        if( key == markers[i].branchKey) {
            return markers[i].location;
        }
    }
    return "location not found";
}

function openInfo(marker, branchKey) {
    var reportSearchPath = '';
    var url = gatewayURL + "/rds/output/searchPath" + reportSearchPath
        + "?fmt=HTMLFragment&p_branch=" + branchKey;

    progressDiv.style.display = "";
    progressDiv.innerHTML= "<h3>Please wait for the info retrieval<p>"+marker.title+"</h3>";

    infowindow.close();
    objXHR.open("GET", url, true);
    objXHR.onreadystatechange = function() {
        if (objXHR.readyState == 4) {
            if (objXHR.status == 200) {
                infowindow.setContent(objXHR.responseText);

                // Tying the InfoWindow to the marker
                infowindow.open(map, marker);
            } else {
```

Also this page requires auto login. (not recommended for the best practice)
Define the following variables:

nameSpace = "OPENLDAP"
username = "team1"
password = "password"

in the following parts:

```
<script>

var gatewayURL = "[illegible]";
var nameSpace = "OPENLDAP";
var userName = "team1";
var password = "password";
var map;
```

```
function displayLogonPage() {
        document.getElementById("gatewayURL").value=gatewayURL;
        document.getElementById("userName").value=userName;
        document.getElementById("nameSpace").value=nameSpace;
        document.getElementById("password").value=password;
        // show the login box
        loginDiv.style.display = "";
}
```

You can test the webpage by the URL:
http://(local ip)/ibmcognos/CMSGoogleMaps.html

Then you have completed this exercise.

IBM Inter-University
Programming Contest 2013

Chapter 3: Tivoli BigFix

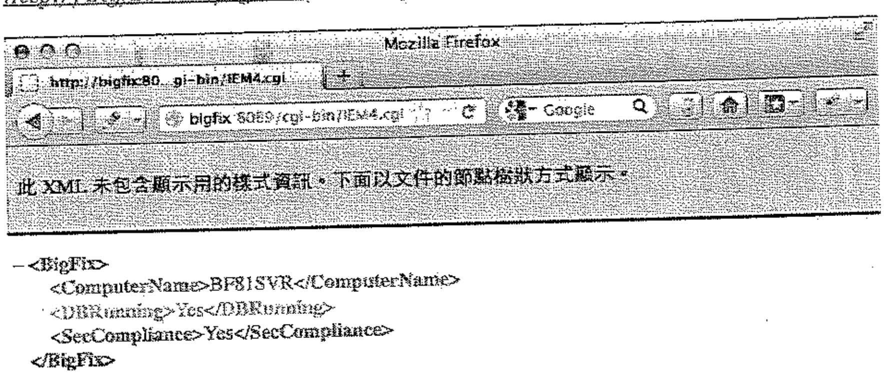
Objectives

In this exercise, we will learn:

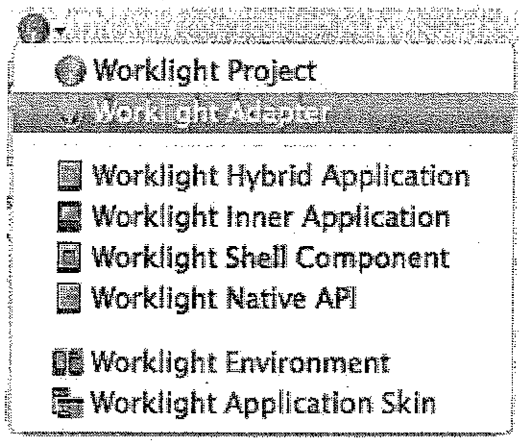
- Review response from BigFix server
- Perform web service request from Worklight to BigFix Server
- Parsing the result and integrate with mobile app

Exercises

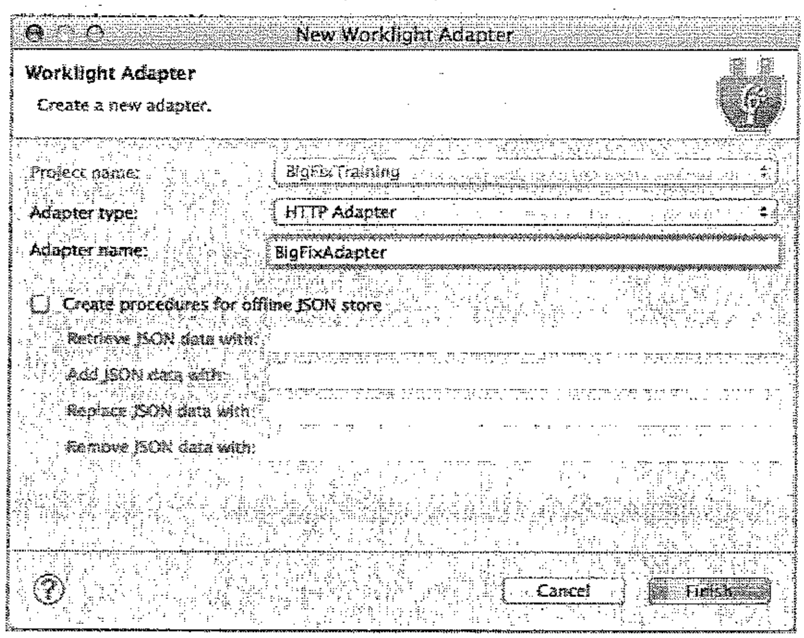
1. Open a Mozilla Firefox browser, and then go to <u>*http://bigfix:8089/cgi-bin/IEM4.cgi*</u>. The following screen is generated:



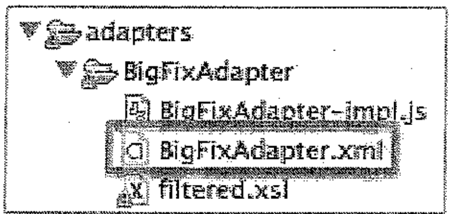
2. Let's make the Worklight enable to communicate with the BigFix server. First, create an HTTP Adapter in the Worklight Project. Find the Worklight button , click the down arrow. Then select "*Worklight Adapter*".



3. Choose the current project (i.e. *BigFixTraining*) in the field "Project name" and choose "*HTTP Adapter*" for the Adapter Type. Enter a name for the adapter, here we will use "*BigFixAdapter*".



4. Configure the adapter with the XML file. Open "*BigFixAdapter.xml*".



5. Define the connectivity settings as follows:

- protocol: http
- domain: bigfix
- port: 8089

```
<connectivity>
    <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
        <protocol>http</protocol>
        <domain>bigfix</domain>
        <port>8089</port>
    </connectionPolicy>
    <loadConstraints maxConcurrentConnectionsPerNode="2" />
</connectivity>
```

6. Declare adapter procedures. Remove all pre-defined procedures and define a new procedure called "*getServerStatus*".

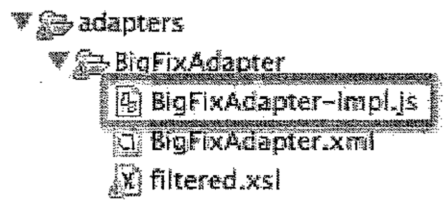
```
<wl:adapter name="BigFixAdapter"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wl="http://www.worklight.com/integration"
    xmlns:http="http://www.worklight.com/integration/http">

    <displayName>BigFixAdapter</displayName>
    <description>BigFixAdapter</description>
    <connectivity>
        <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
            <protocol>http</protocol>
            <domain>bigfix</domain>
            <port>8089</port>
        </connectionPolicy>
        <loadConstraints maxConcurrentConnectionsPerNode="2" />
    </connectivity>

    <procedure name="getServerStatus"/>

</wl:adapter>
```

7. Implement procedures in the JavaScript file. Open "*BigFixAdapter-impl.js*".

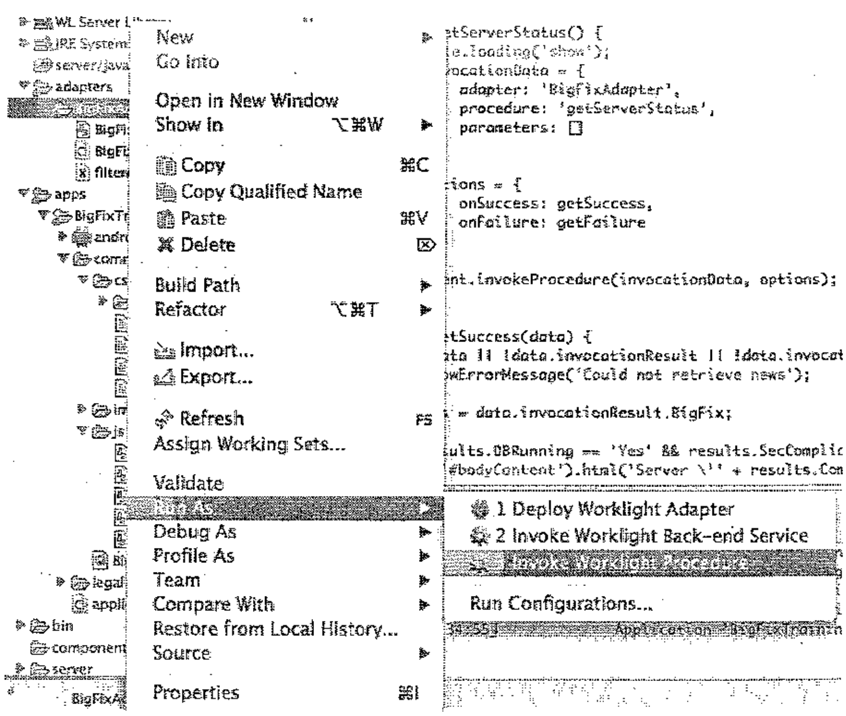


8. Remove all code and insert the following:

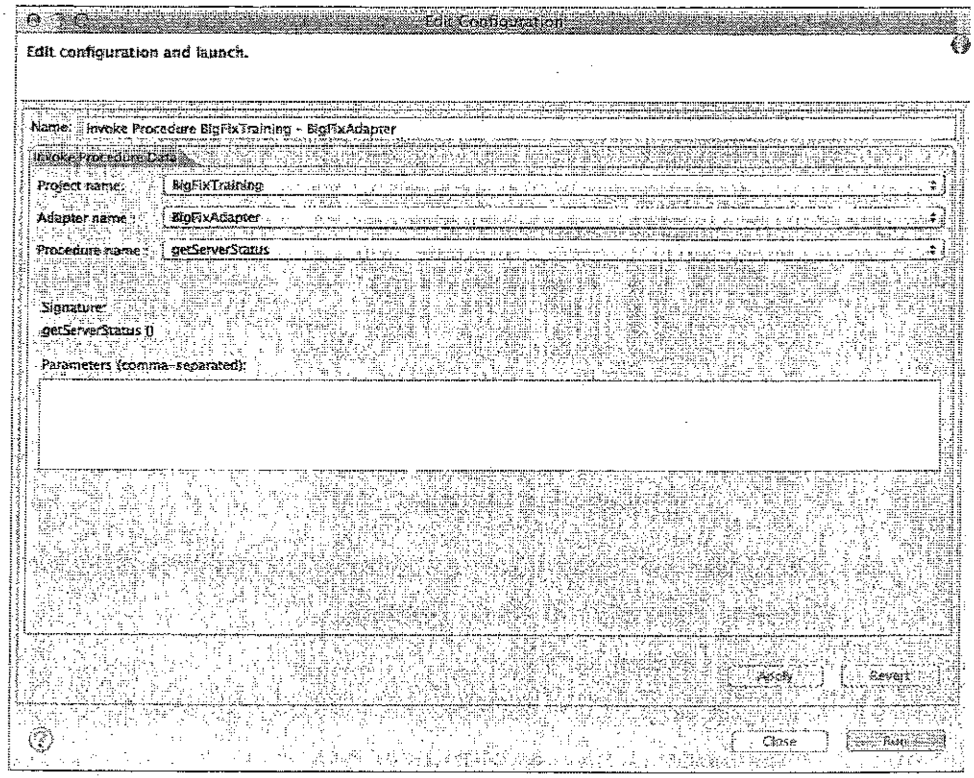
```
function getServerStatus() {
    var input = {
        method : 'get',
        returnedContentType : 'xml',
        path : 'cgi-bin/IEM4.cgi'
    };

    return WL.Server.invokeHttp(input);
}
```

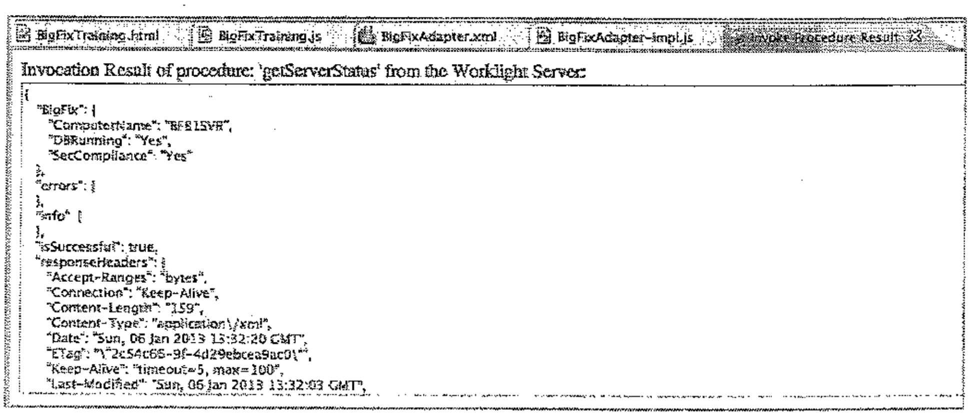
9. Now, try if the adapter can retrieve response from BigFix Server.



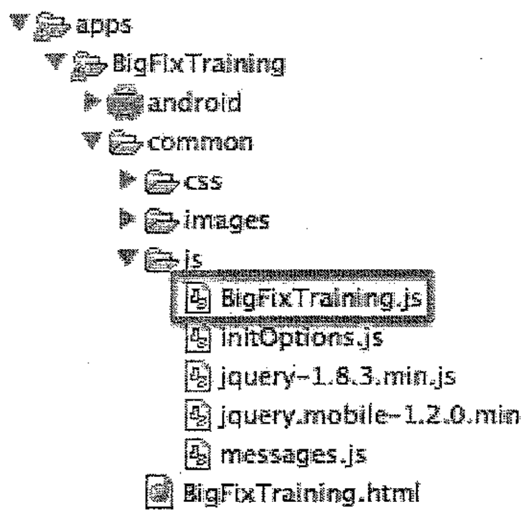
10. Choose the corresponding project name (i.e. "*BigFixTraining*") and adapter name (i.e. "*BigFixAdapter*"). Choose "*getServerStatus*" as the procedure name to invoke. Keep parameters empty and click "*Run*".



11. The result should be somehow in the following format.



12. Then you can go on to integrate the results into the mobile application. The application layout interface has already been provided. You should write code to display the results. Open "*BigFixTraining.js*".



13. In the *getServerStatus()* function, create an *invocationData* object for the procedure call with the following format:

```
var invocationData = {
        adapter: 'adapterName',
        procedure: 'procedureName',
        parameters: []
};
```

Fill in corresponding information for the procedure call.

14. Create an *options* object to define success and failure handlers.

```
var options = {
        onSuccess: getSuccess,
        onFailure: getFailure
};
```

15. Create a Worklight Procedure call.

```
WL.Client.invokeProcedure(invocationData, options);
```

16. In the *getSuccess(data)* function, write code to display conditional content depending on the result from BigFix server. The returned *BigFix* object has been stored to a variable called *result*. You can access object properties *ComputerName, DBRunning* and *SecCompliance* by calling *result.ComputerName, result.DBRunning* and *result.SecCompliance* respectively. The following is an example to display the sentence "The quick brown fox jumps over the lazy dog":

```
$('#bodyContent').html('The quick brown fox jumps over the lazy
dog');
```

17. Now you can test the application.

IBM Software
WebSphere

Lab:

Getting Started with IBM Worklight

Lab Exercise

IBM

Deleted: 1

Contents

Deleted: 17

Deleted: 19

Deleted: 25

Deleted: 48

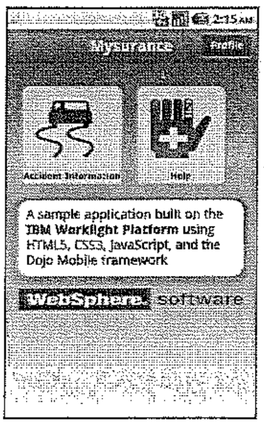
Deleted: 52

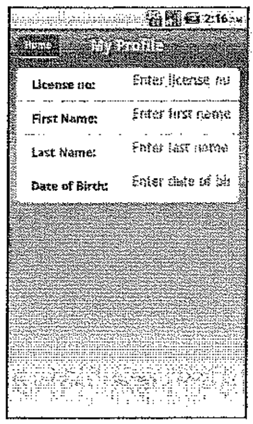
Lab 1 Getting Started with IBM Worklight

In this lab you will develop a basic mobile app using the IBM Worklight Studio development environment. You will use cross platform techniques such as HTML5, CSS3, JavaScript and the jQuery Mobile framework. The app you will develop and use throughout this Lab is called Mysurance and its purpose is to help create an insurance claim in case you have a car accident, and also to help you out while at the accident scene. Below are some screenshots from the fully developed application:

Deleted: JQuery

Deleted: PoT





In this lab you will build a simplified version of these two views in the rich page editor.

Upon completion of this exercise you should have gained basic understanding of

- How to create a Worklight project and a Worklight application in Worklight Studio
- How to build and deploy a Worklight application to the test server in the Worklight Studio
- How to use the Mobile Device Simulator to preview and test an application from the Worklight Studio
- How to create Worklight Environments for platforms such as Android, iPhone, etc.
- How to use the Rich Page Editor to add UI elements to an application

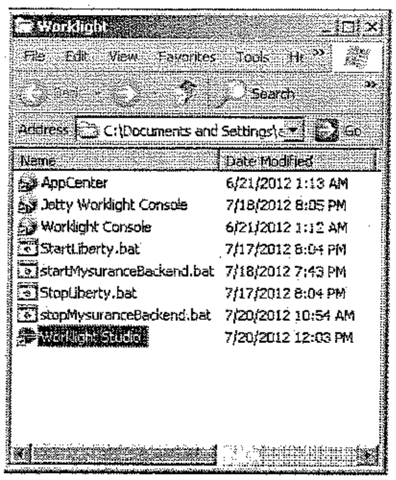
You should possess basic knowledge of HTML, CSS and JavaScript. Familiarity with the Eclipse platform is a plus, but not required.

1.1 Start the Worklight Studio

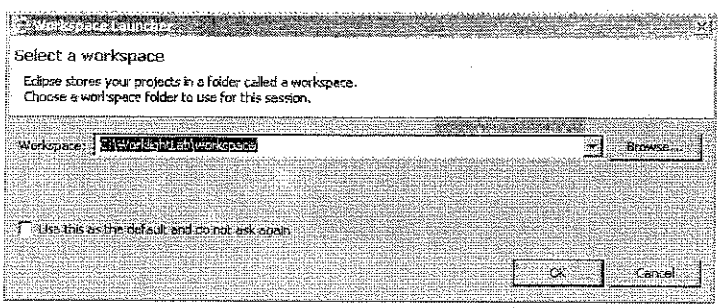
This lab assumes that you have obtained and started the corresponding VMWare image. In the image you will launch Eclipse with the Worklight Studio tooling and then create a project for the Mysurance app.

__1. Start Worklight Studio

__a. In the Worklight folder double-click the **Worklight Studio** icon.



__b. On the Workspace Launcher dialog accept the default workspace path **C:\WorklightLab\workspace** and click **OK**.

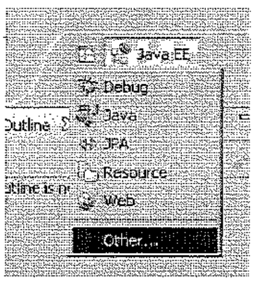


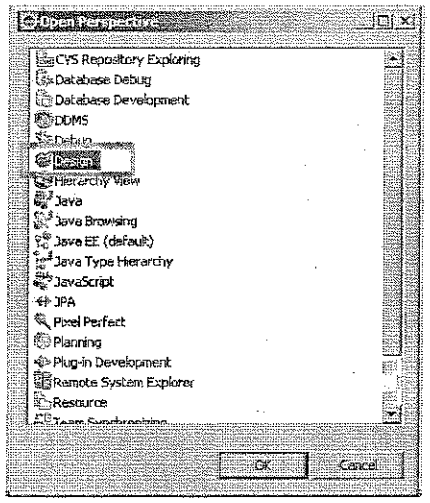
__c. If you receive an Eclipse Welcome Screen, dismiss it by **closing** the *Welcome* tab.

1.2 Create the Mysurance project and application

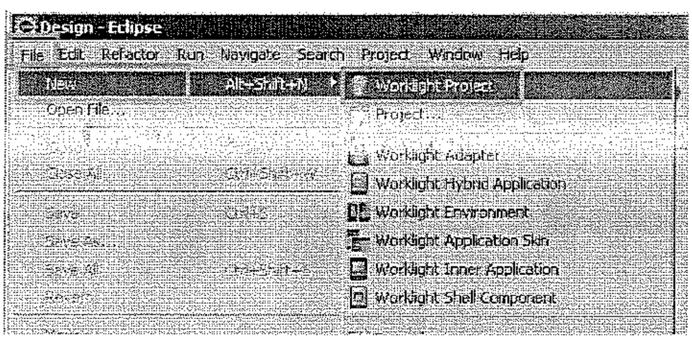
__1. Create Mysurance project

__a. Switch to the Design perspective by clicking the **Open perspectives icon** and selecting **Other > Design**.

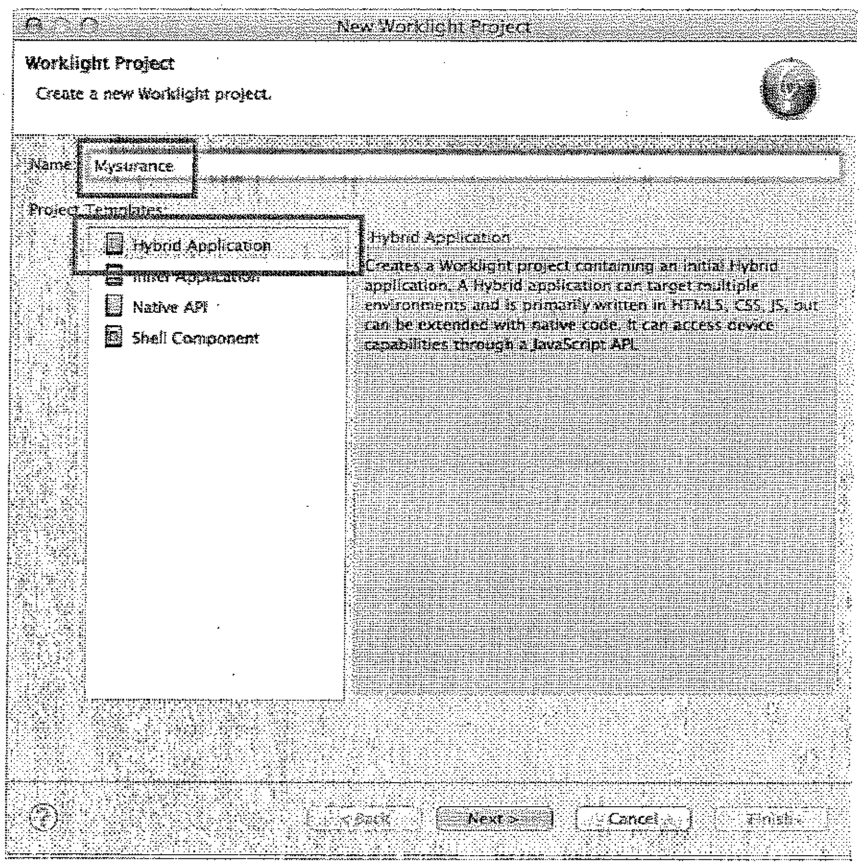




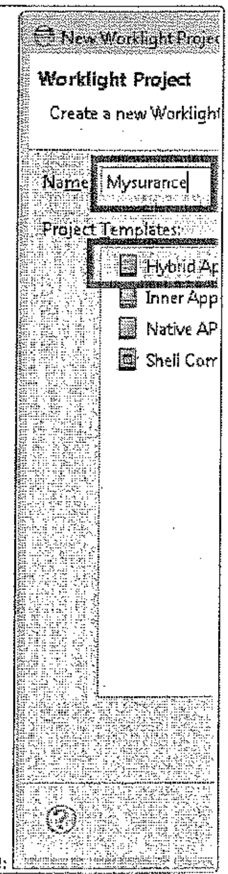
__b. Select **File > New > Worklight Project**.



__c. Enter **Mysurance** as the project name, keep the default Project Templates selection for **Hybrid Application** and click **Next**.



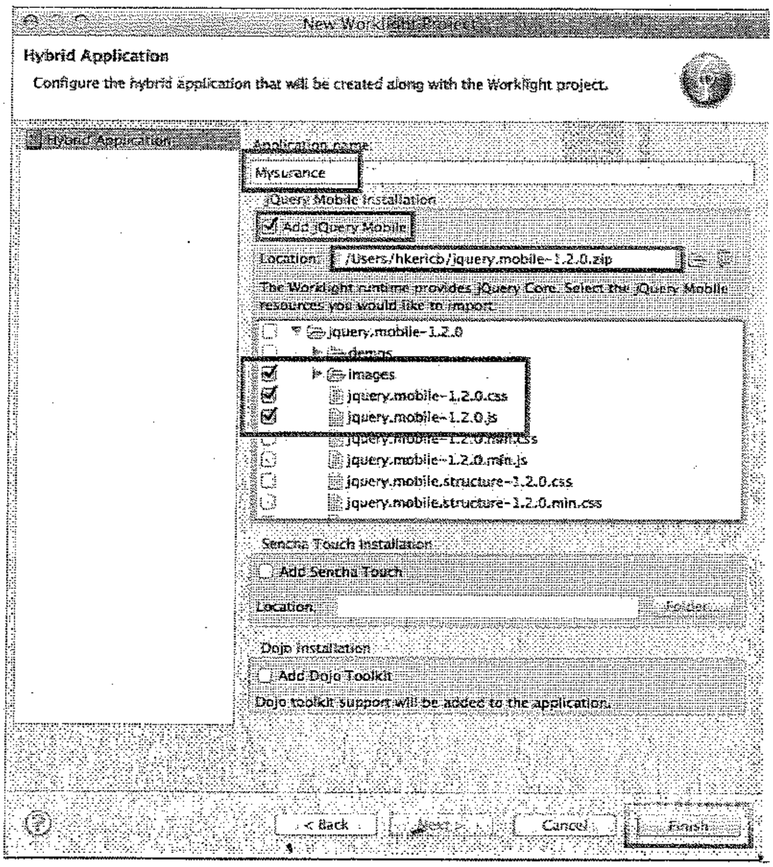
Deleted:



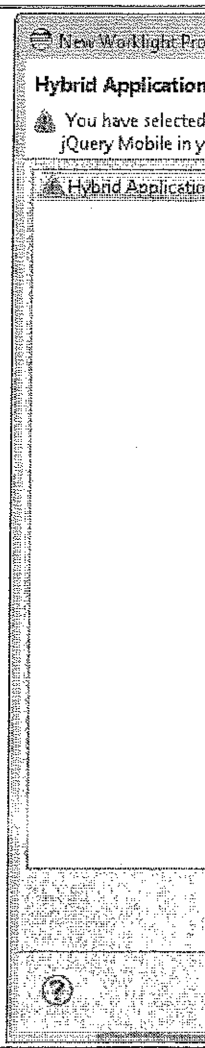
__d. The Hybrid Application panel will surface noting that you have not specified an Application name. Enter **Mysurance** for the Application Name, check the box to **Add jQuery Mobile**, enter the **Location**, check the images folder, CSS and JavaScript files of jQuery Mobile, i.e. **jquery.mobile-1.2.0.css and jquery.mobile-1.2.0.js**. Click **Finish**

Formatted: Font: Bold

Formatted: Font: Not Bold

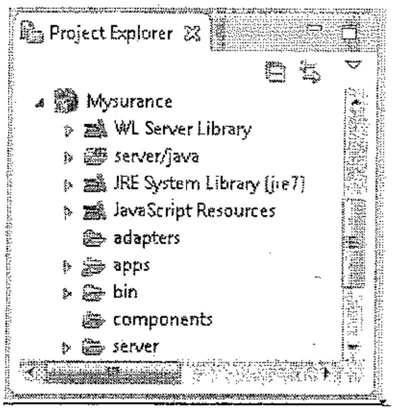


Deleted:

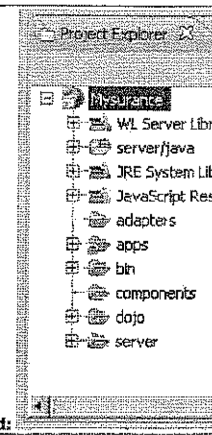


__e. The application template will be populated and the **application-descriptor.xml** file will open by default. Application characteristics such as authentication and server URL are managed in this file. We can leave it at its defaults for now, while we investigate the parts of a Worklight project and application.

__f. In the *Project Explorer* pane, expand the **Mysurance** project. Review the folder structure that has been created.



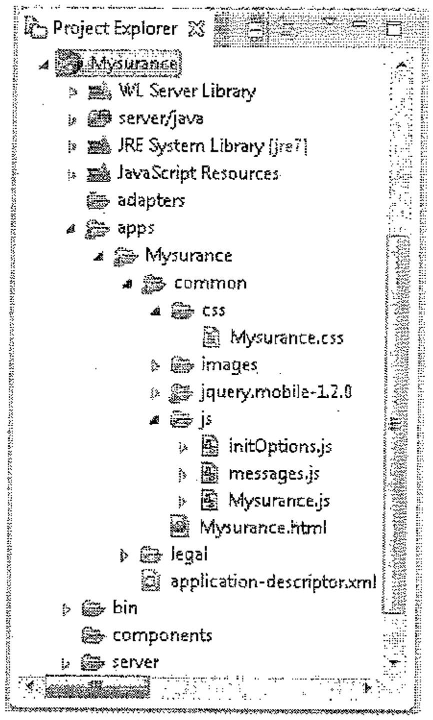
WL Server Library
Contains the Worklight API jar file
server/java
Location for server-side java code in java-base adapters (advanced)
JRE System Library
Contains the JRE used in this project
JavaScript Resources
Contains the project's JavaScript classes content
adapters:
Contains the project's adapters (used for backend connectivity)
apps:
Contains the project's applications
bin:
Location for build artifacts (wlapp files) that are deployed to a Worklight server
components
Contains shell application components (advanced)
Server
Contains configuration files and extension locations for the embedded Worklight test server



Deleted:

Deleted: dojo¶
Contains the Dojo Toolkit JavaScript source¶

__g. In the *Project Explorer*, expand the **apps** folder then the **apps > Mysurance** folder and the apps > Mysurance > common folder that were created by the new application wizard.



common: the default 'environment' that gets created for an application.

css: Mysurance.css – the main application CSS file, **reset.css** – brings all rendering engines to one common ground.

images: Default images for the common environment.

js: Mysurance.js – the main JavaScript file for the app, **messages.js** – JSON object holding all app messages, **initOptions.js** – init options for the app.

Mysurance.html: The main application html file. Application can have multiple html files

legal: All legal related documents.

application-descriptor.xml: Application's meta data (security config, server url, etc...)

__h. Open the **application-descriptor.xml** file, if not already opened. Switch to the **Source** tab. The following section specifies the application name, description and author's name to be displayed in the Worklight Console.

application-descriptor.xml

```
<?xml version="1.0" encoding="UTF-8"?>

    <!-- Licensed Materials - Property of IBM
         5725-G92 (C) Copyright IBM Corp. 2006, 2012. All Rights Reserved.
         US Government Users Restricted Rights - Use, duplication or
         disclosure restricted by GSA ADP Schedule Contract with IBM Corp. -->

<!-- Attribute "id" must be identical to application folder name -->
<application id="Mysurance" platformVersion="5.0.5"
    xmlns="http://www.worklight.com/application-descriptor"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <displayName>Mysurance</displayName>
    <description>Mysurance</description>
    <author>
        <name>application's author</name>
        <email>application author's e-mail</email>
        <copyright>Copyright My Company</copyright>
        <homepage>http://mycompany.com</homepage>
    </author>
```

Design Source

i. The **worklightRootURL**, is to be used as the root URL in generated mobile applications.

application-descriptor.xml

```
<worklightServerRootURL>http://${local.IPAddress}:8080</worklightServerRootURL>
<!--
    Uncomment one of the following options to use a different value for the Worklight Server root URL
    <worklightServerRootURL>http://${local.hostname}:8080</worklightServerRootURL>
    <worklightServerRootURL>http://${local.fullhostname}:8080</worklightServerRootURL>
    <worklightServerRootURL>http://${local.IPAddress}:8080</worklightServerRootURL>
-->
```

Design Source

j. Environment specific information will be inserted automatically as new 'environments' are added to the project. You can observe this change as environments are added during later portions of the lab.

Deleted: <#>The following section specifies authentication details. This indicates whether and when the app requires users to authenticate. Possible values include: ¶
never: Users are never required to authenticate¶
onStartup: Users are required to authenticate before using the app¶
onDemand: The app will trigger login process if the developer triggers it explicitly, and when it determines that authentication is required.¶

application-descriptor.xml

```
<usage requireAut
<!--
    Uncomment the
    <authenticati
        <device :
    </authenticat
-->
<!--
    Uncomment one
    <usage requir
    <usage requir
-->
```

Design Source

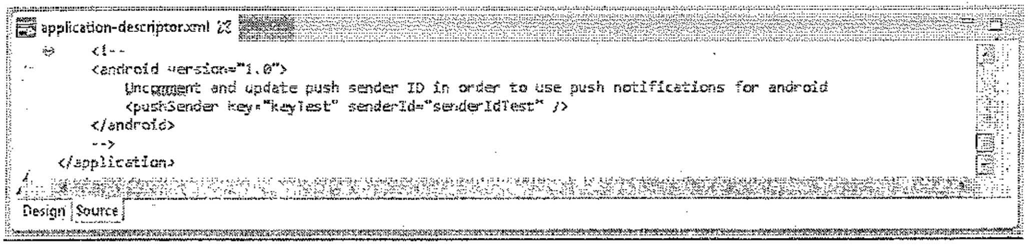
Formatted: Bullets and Numbering

application-descript

```
<worklight
<!--
    Uncomm
    <workl
    <workl
    <workl
-->
```

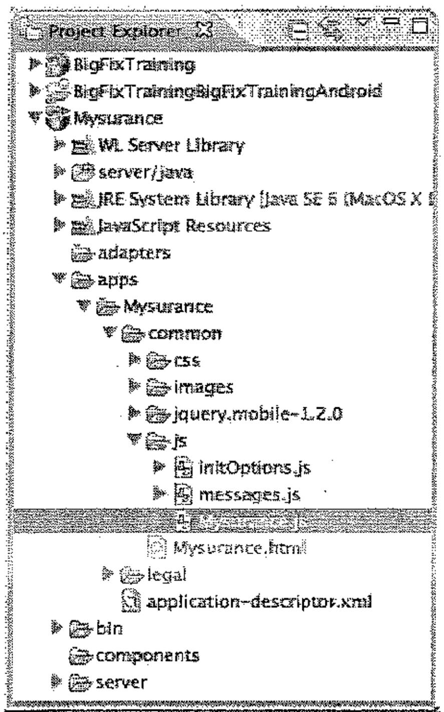
Deleted: Design Source

Formatted: Bullets and Numbering

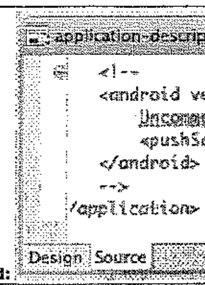


__2. Add logic to fix the jQuery Mobile bug.

__a. In the *Project Explorer*, locate and open the **Mysurance.js** file under Mysurance > apps > Mysurance > common > js



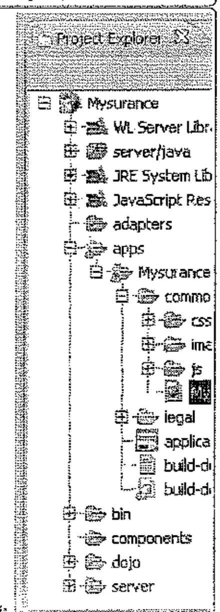
__b. After the line "`// Common initialization code goes here`", enter "`$('#content').resize();`" (without quote). The resulting code should look like this:



Deleted:

Deleted: Add logic to handle a connection failure with the Worklight server

Deleted: Mysurance.html



Deleted:

Deleted: During the runtime of an application, the main HTML document cannot be replaced by another HTML document. The Worklight Client framework initialization is boun ... [1]

Formatted: Bullets and Numbering

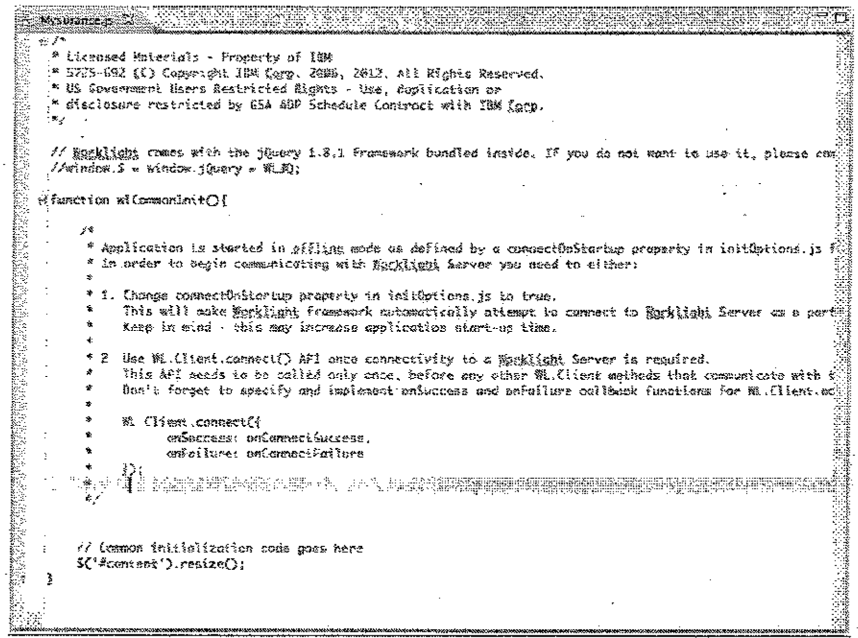
Formatted: Font: Courier

Formatted: Font: Courier

Deleted: In the init method, enter "`timeout:2000, onConnectionFailure: doConnectionFailure`" (without the quote) inside the onload me ... [2]

Formatted: Font: 10 pt

Deleted: <body> tag



```
/*
 * Licensed Materials - Property of IBM
 * 5725-G92 (C) Copyright IBM Corp. 2006, 2012. All Rights Reserved.
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
 */

// Worklight comes with the jQuery 1.8.1 framework bundled inside. If you do not want to use it, please com
//window.$ = window.jQuery = WLJQ;

function wlCommonInit(){

	/*
	 * Application is started in offline mode as defined by a connectOnStartup property in initOptions.js f
	 * In order to begin communicating with Worklight Server you need to either:
	 *
	 * 1. Change connectOnStartup property in initOptions.js to true.
	 *    This will make Worklight framework automatically attempt to connect to Worklight Server as a part
	 *    Keep in mind - this may increase application start-up time.
	 *
	 * 2  Use WL.Client.connect() API once connectivity to a Worklight Server is required.
	 *    This API needs to be called only once, before any other WL.Client methods that communicate with t
	 *    Don't forget to specify and implement onSuccess and onFailure callback functions for WL.Client.co
	 *
	 *    WL.Client.connect({
	 *    		onSuccess: onConnectSuccess,
	 *    		onFailure: onConnectFailure
	 *    });
	 *
	 */


	// Common initialization code goes here
	$('#content').resize();
}
```

__c. **Save** the Mysurance.js file.

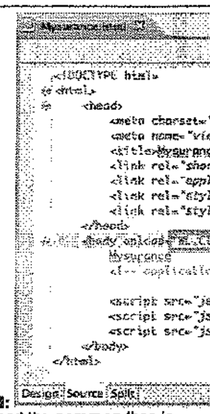
__3. Perform a build of the work that you have completed so far.

__a. Right-click the **Mysurance** application and select **Run As > Build All and Deploy.**

Deleted: ¶
<body onload="WL.Client.init({ timeout:2000 , onConnectionFailure: doConnectionFailure })" id="content" style="display: none">¶
Use above or Snippet #1 from the accompanying lab1-snippets.txt file for cut & paste.¶
The WL.Client can accept a number of parameters. First, we set the connection timeout value to be 2 seconds. Secondly, to add connectivity failure detection in the init method, we added onConnectionFailure property in the onLoad method and specified a callback function named doConnectionFailure to be invoked in case connectivity fails.

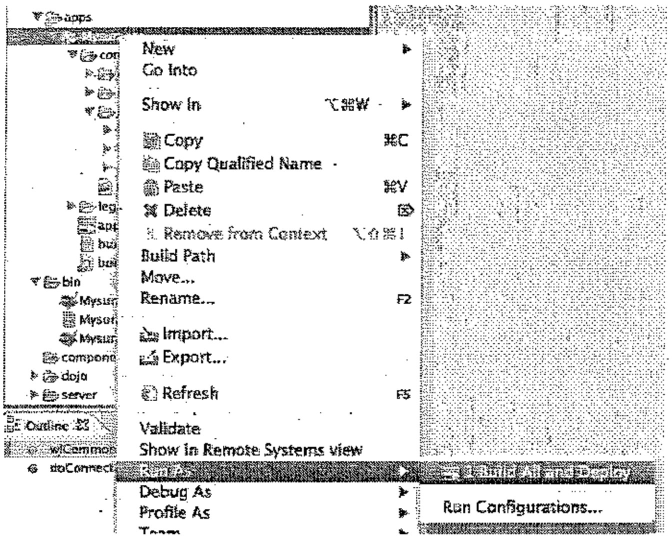
Formatted: Indent: First line: 0"

Deleted: Mysurance.html



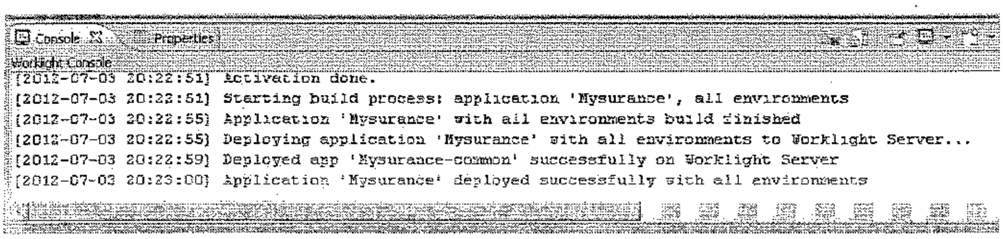
Deleted: <#>Expand the **common** then **js** folder, open the **Mysurance.js** file. This is the application's main JavaScript file. It contains the wlCommonInit() function that will be invoked automatically once the Worklight framework initialization completes. You can add your application's initialization code ... [3]

Formatted: Bullets and Numbering



The Build and Deploy step will publish your application to an embedded test server within eclipse, where we can preview and test as part of the development life-cycle.

__b. Verify that the build process and that the deployment to Worklight server was successful by examining the log output in the Console view.

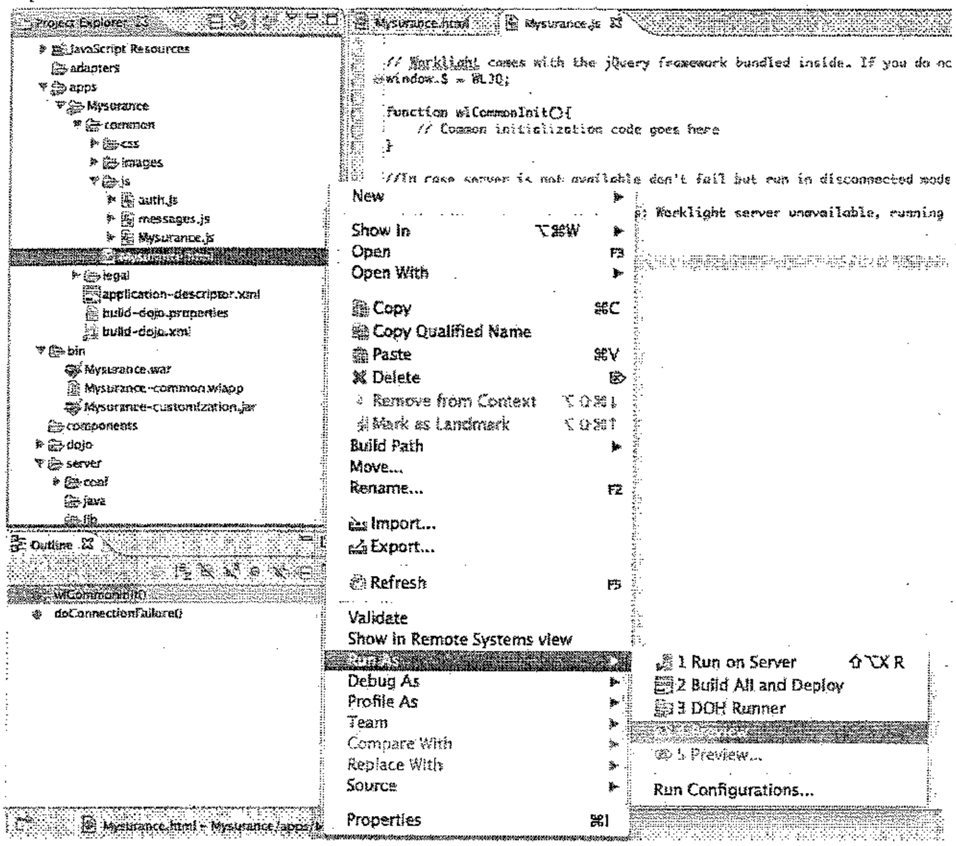


If the console is not displayed, open it with **Window > Show view > Console.**

Use Window > Use the console icon ( ) to switch between the various consoles if necessary, until you find the **Worklight Console**.

1.3 Preview common resources in Mobile Browser Simulator

__1. Right-click on either the **common** folder (to run the common resource web app) or the **mysurance.html** file within the common folder and select **Run as > Preview** (use the first preview option in the menu)



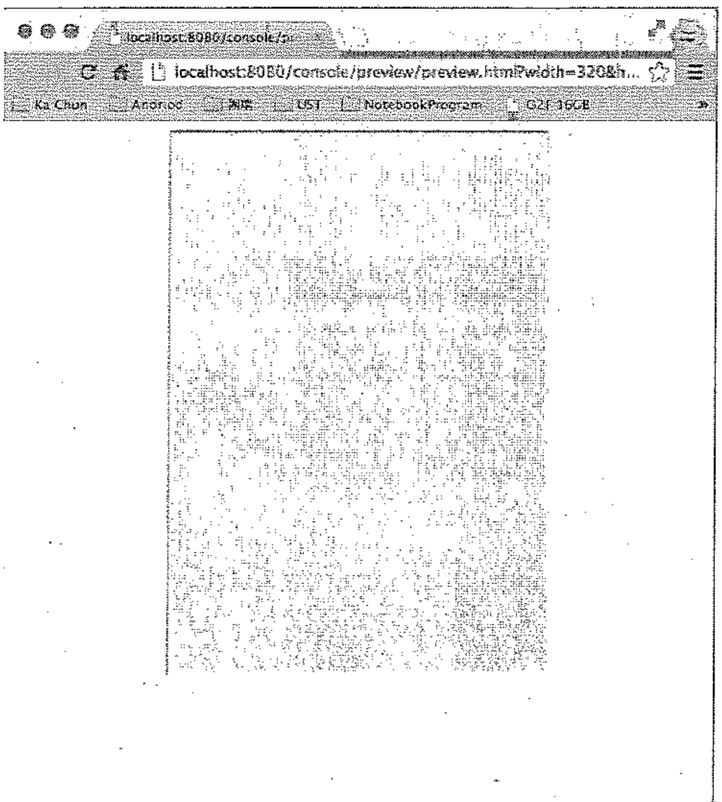
__2. A new external browser window will open with the **Mobile Browser Simulator** rendering our Mysurance application.

For this lab, we have configured eclipse to use an external web browser (Chrome), which opens the Mobile Browser Simulator in a browser window outside of eclipse. Using an external browser gives freedom to choose your rendering browser, access to advanced debugging features like Web Inspector in Chrome or Firebug with Firefox, and access to internet settings, cache and history that are not available when running with the internal browser setting in Eclipse (see Window > Preferences > General > Web Browser for this setting).

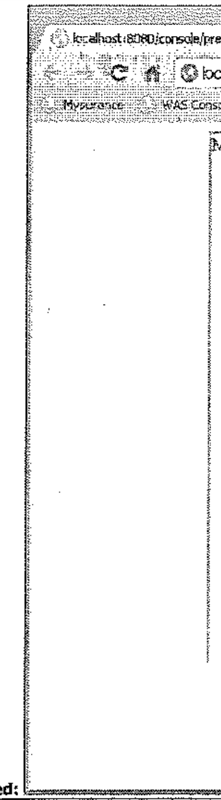
Deleted: eclipse

As this application has no device characteristics yet, a simple, mobile browser view is rendered, with no device configuration options.

Deleted: un-styled



Deleted:



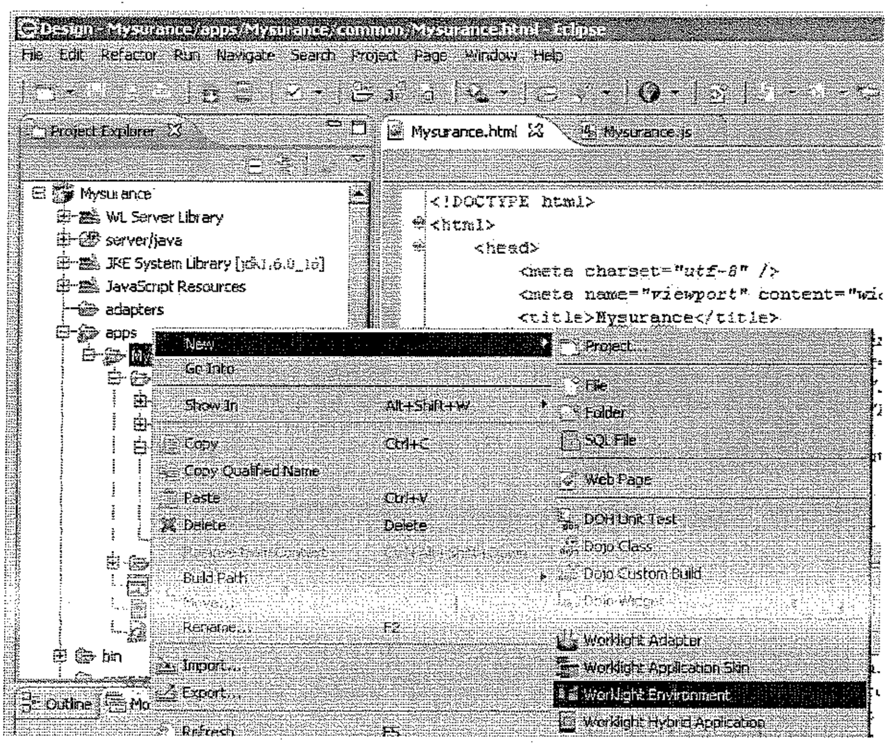
Congratulations! You are previewing your first hello world-like application developed using Worklight.

1.4 Exploring Worklight Environments

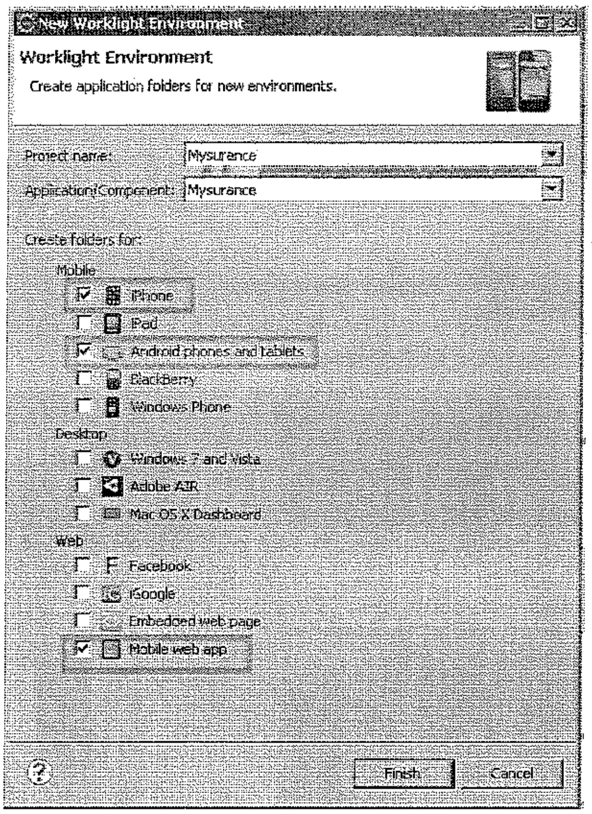
An environment is a mobile, desktop, or web platform capable of displaying web-based applications, such as the Apple iPhone, Android phones, Blackberry, Windows Phone and Windows 8. In this section you will create environments to provide support for iPhone, Android and Mobile Web.

__1. Creating the Worklight Environments for the Mysurance application

__a. In the *Project Explorer* select the **Mysurance** application (in the /Mysurance/apps folder). In the right click menu select **New > Worklight Environment.**

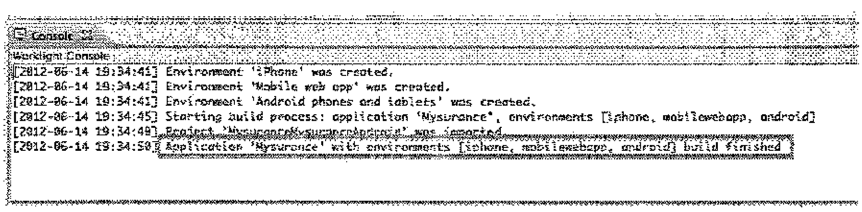


__b. In the *Worklight Environment* dialog, select **iPhone, Android phones and tablets,** and **Mobile web app**, then click **Finish**.

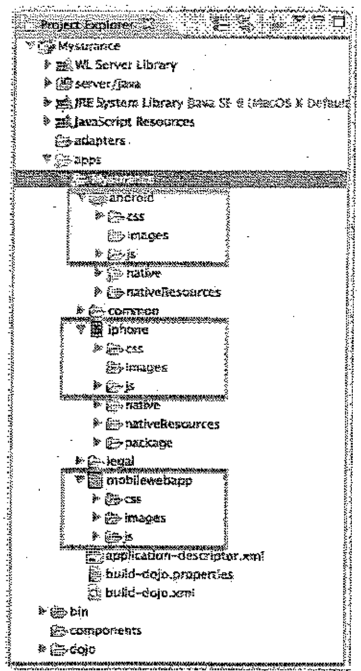


Note: In previous versions of the VMware image, we have seen an assertion failure here. If you see this, please re-try.

__c. Observe the *Console* window, notice that the messages about the environments that you have just chosen.



__d. In the *Package Explorer*, observe that there are now additional folders created under the **Mysurance** application folder.

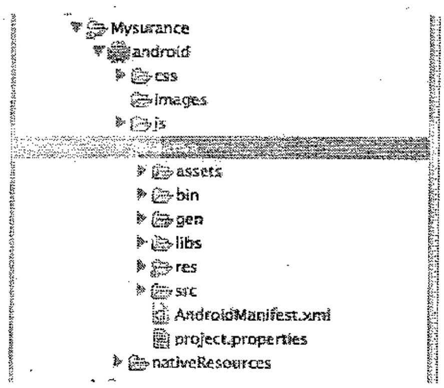


The new environment's resources will have the following relationship with the common resources:

- **images** - override the common images in case both share the same name
- **css** – extend and/or override the common CSS files
- **js** - extends the common application instance JS object (The environment class extends the common app class)
- **HTML** - override the common HTML code in case both share the same name
- **native:** contains environment specific generated application code

__2. Quick review of the native folder for specific environments

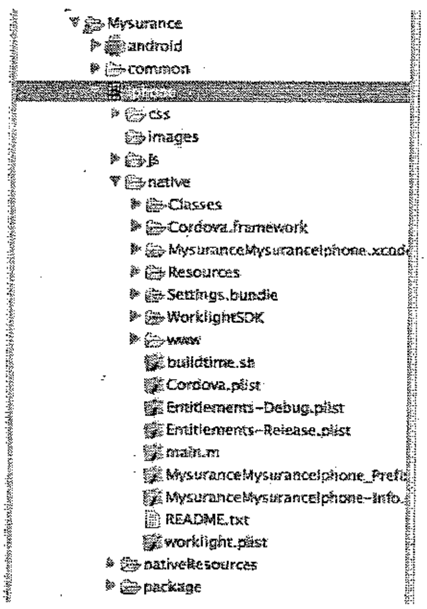
__a. Android native folder



The **native** folder under android contains automatically generated android application code that is imported into the eclipse workspace as an Android Project during the Build and Deploy step (later).

It is not recommended to edit files under the assets folder, as each time the application is built they are regenerated.

__b. iPhone folders



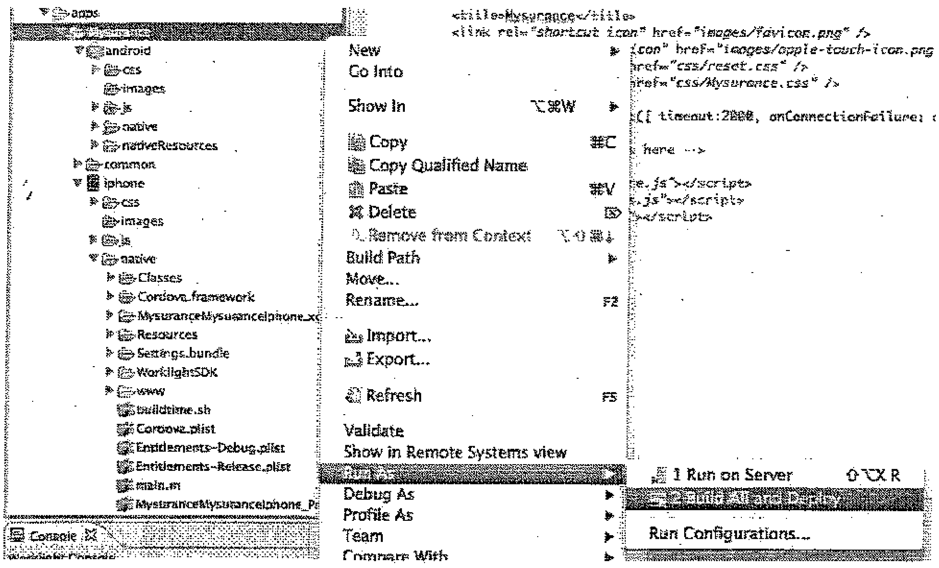
The **native** folder under iphone contains automatically generated iphone app code

The **package** folder under iphone contains a packaged (zipped) application

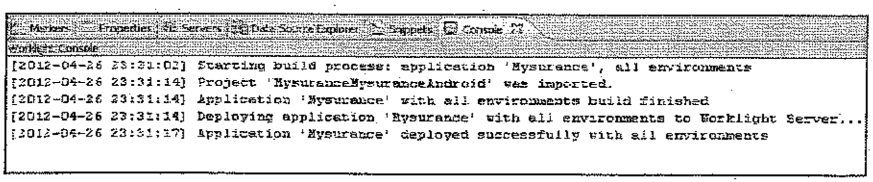
It is not recommended to edit files under native/www folder, as each time the application is built they are regenerated

__3. Build and deploy all environments

__a. Right-click the **Mysurance** application and select **Run As > Build All and Deploy.**

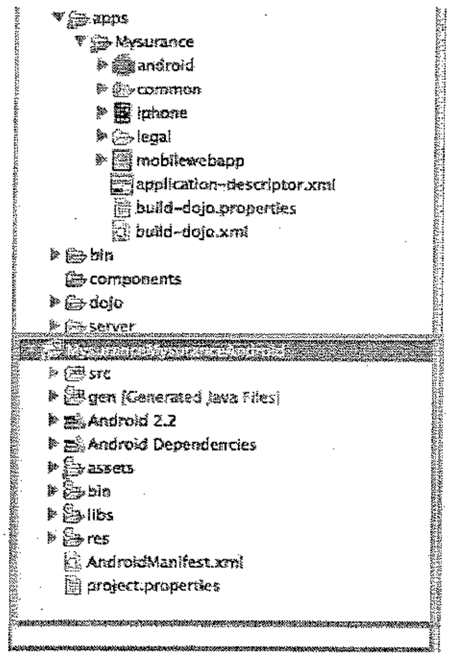


__b. Check the Console log to make sure that the build and deploy is successful.



__c. Observe that the Native Android project has been created and imported into Eclipse workspace.

This is an example of IBM Worklight's tight integration with the device SDKs. Because the Android SDK is Eclipse-enabled, Worklight is able to immediately generate an Android Project in the Eclipse workspace (same for BlackBerry). For other platforms, Worklight will launch the respective non-Eclipse tooling and provide a project in that tool's format for completion (eg. Xcode).



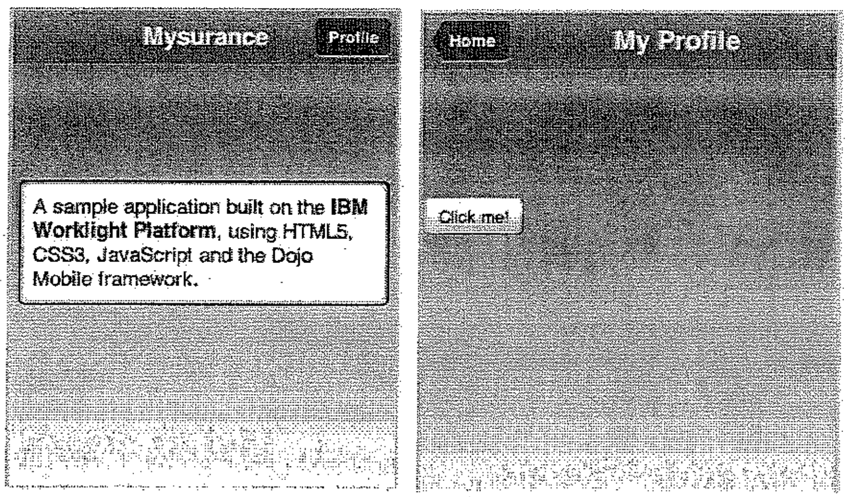
We will add some more interesting content in the next section before previewing the environment-specific variations of the application.

1.5 Enhancing the Mysurance application with visual content

In this part of the Lab, we are going to continue developing our Mysurance application with the new Rich Page Editor, added in Worklight v5.0.

__1. Review the content you will build - two Mysurance application views

__a. The Main view and Profile view of the Mysurance application that will be built in this lab are shown below:



The Main view consists of a jQuery Mobile page, with a header, a navigation button to go to the Profile view, and some sample text.

The profile view is also built on a jQuery Mobile page, and contains a header, a navigation button to return to the Main view, and a button to use for launching a pop-up.

Deleted: dojox.mobile.ScrollableView

Deleted: Header

Deleted: dojox.mobile.ScrollableView

Deleted: Header

__2. Prepare Eclipse for Rich Page Editor development by opening the appropriate Eclipse Views.

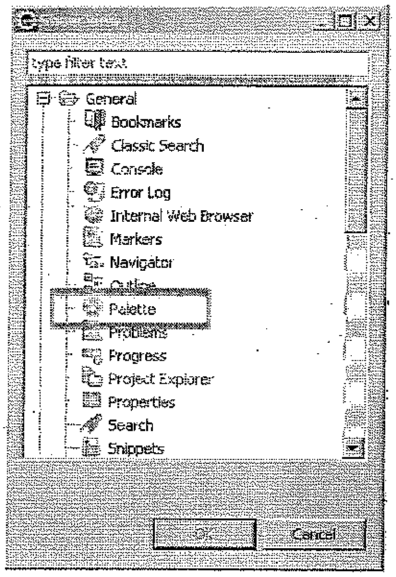
Ensure that you have both the **Palette** and **Mobile Navigation** tabs visible for use with the Rich Page Editor. We will use Mobile Navigation to manipulate and navigate between the Page Views in Mysurance.html, and the Palette will allow us to drag visual elements onto the page.

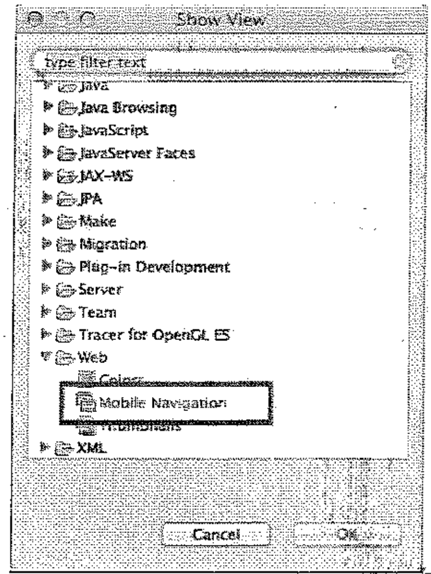
Deleted: eclipse

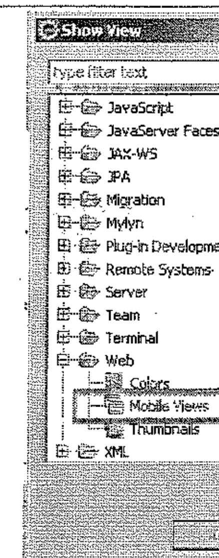
Deleted: Views

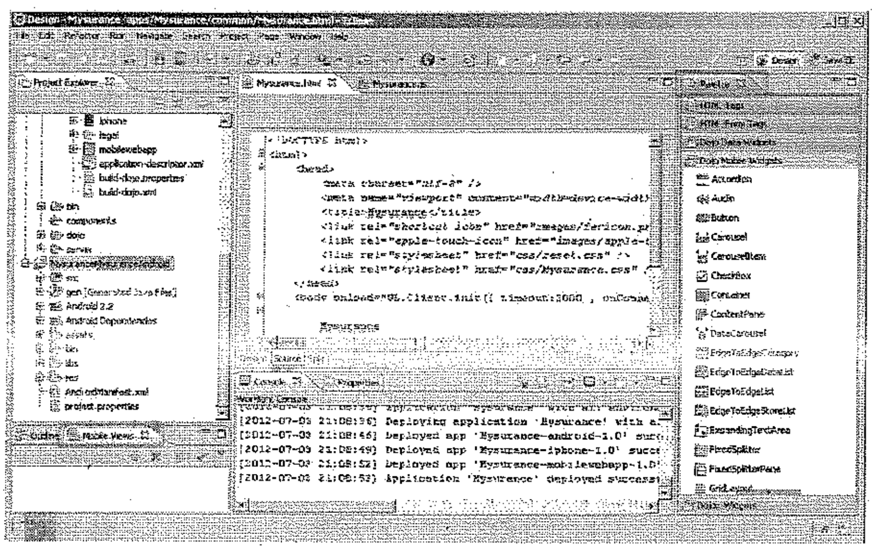
Deleted: Views

Deleted: ScrollableViews





Deleted: 



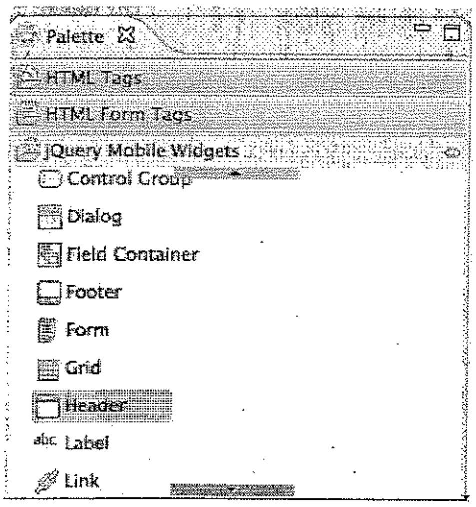
(You can place the tabs in whichever way seems most efficient to you)

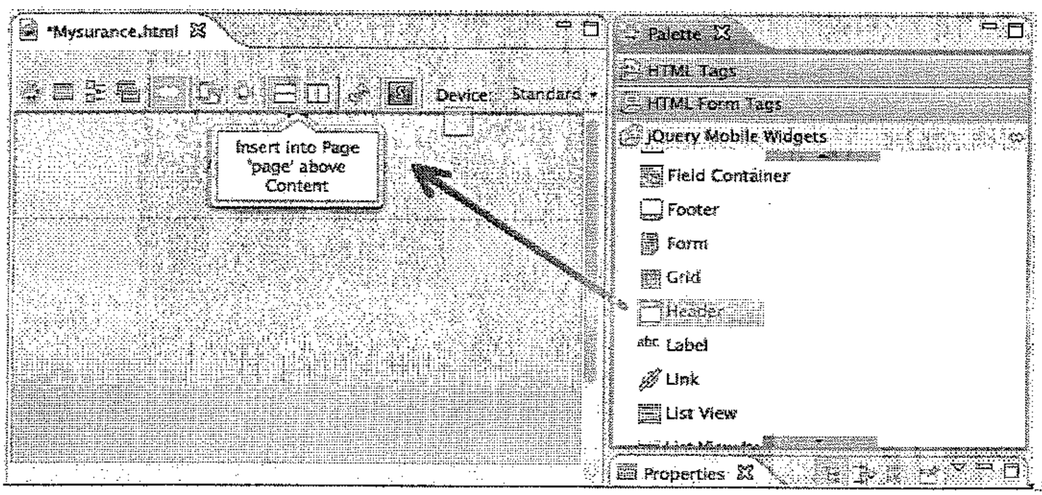
__3. Open the file Mysurance.html into the editor if it's not already opened.

__4. Add a Header to the Mysurance.html page.

Deleted: ScrollableView

Drag the **Header** widget from the **jQuery Mobile Widgets** section of the palette into the design tab of the editor, in the `<div data-role="page" id="page">` section of the Mysurance.html file before the `<div data-role="content" style="padding: 15px">` section.





Deleted: "Pages" in a mobile application are built on views. For this application we are using dojox.mobile.ScrollableViews, which are dojo view widgets that manage a view pane. Ensure that the Mysurance.html file is the active file in the editor pane, and that the de ... [4]

Deleted: ScrollableView

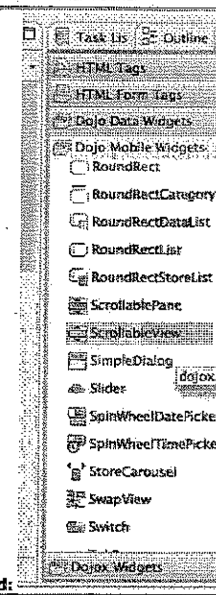
Deleted: Dojo

Deleted: <body>

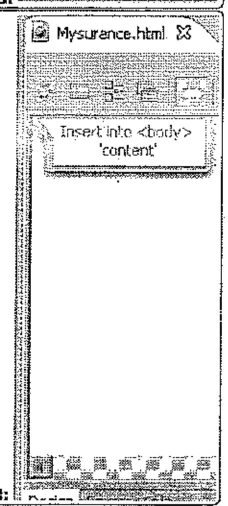
Formatted: Font: Courier

Formatted: Font: Courier

Deleted:



Deleted:

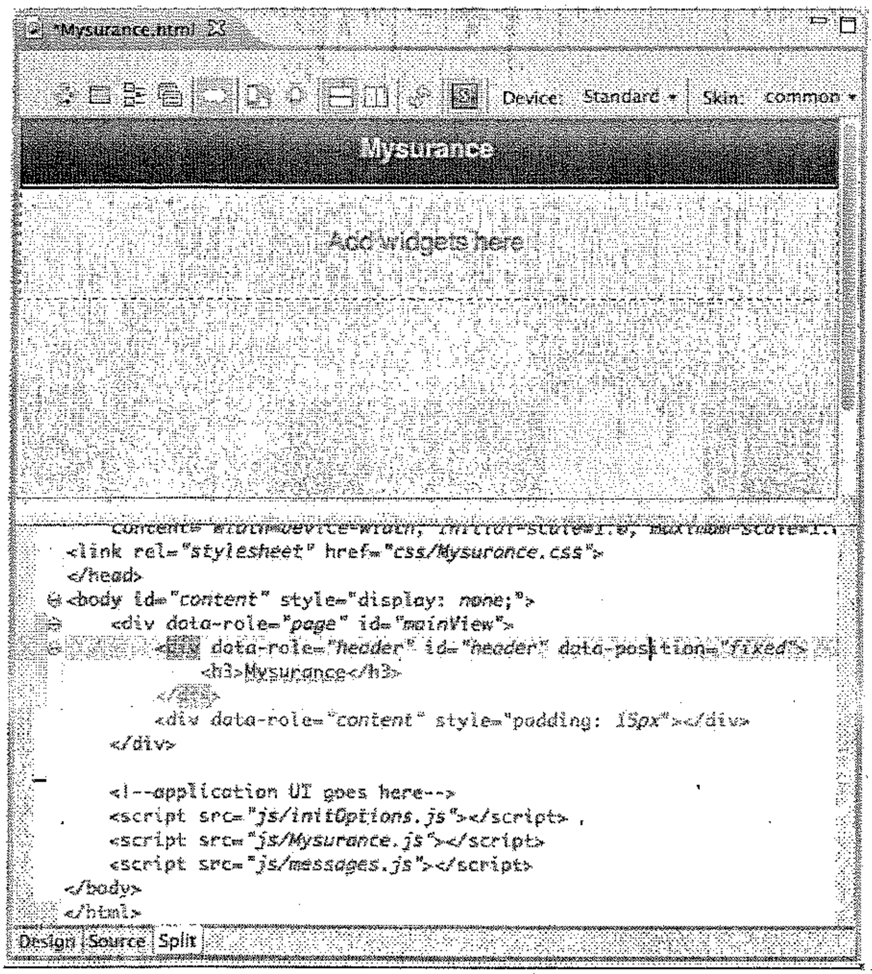


__5. Modify the HTML code of the page as follows.

__a. Change the `id` attribute of the page view from "page" to "mainView".

__b. Change the heading label from "Header" to "Mysurance".

__c. Remove the default "Mysurance" text that now sits below the page `<div>` by deleting it from the source pane in the Rich Page Editor pane.



Deleted: Complete the 'Create a new Dojo Mobile Scrollable View' panel

Deleted: Edit

Deleted: `Id`

Formatted: Font: Courier

Deleted: field

Deleted: and change the default name

Deleted: view0

Deleted: of

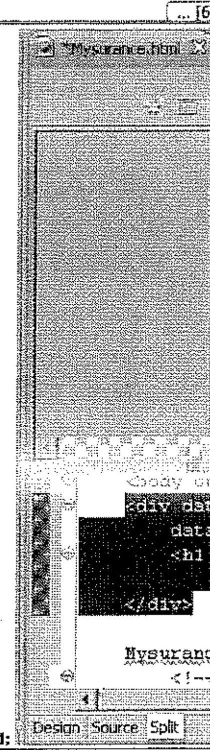
Deleted: Check Set as default view checkbox

Formatted ... [5]

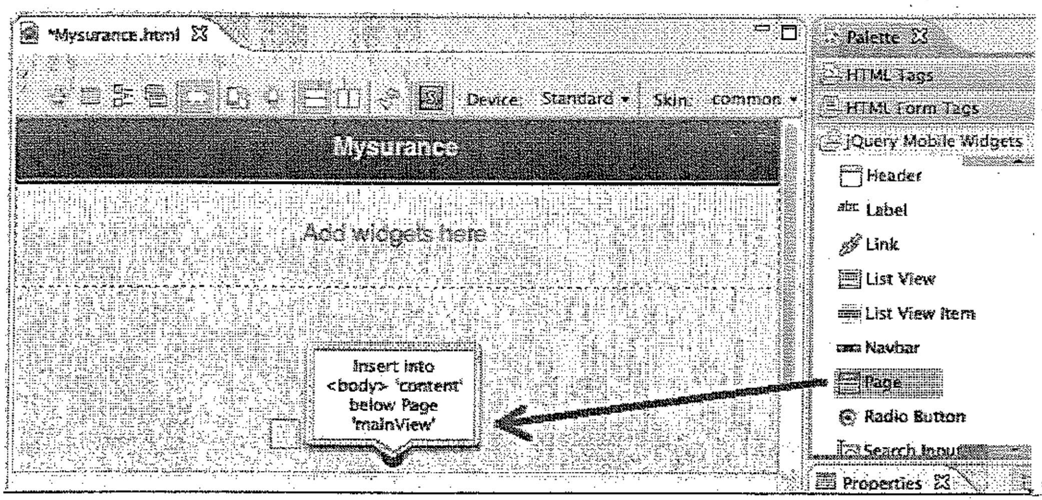
Formatted: Font: Courier

Deleted: Select the Include heading checkbox.¶
<#>Enter `Mysurance` as the heading label.¶ ... [6]

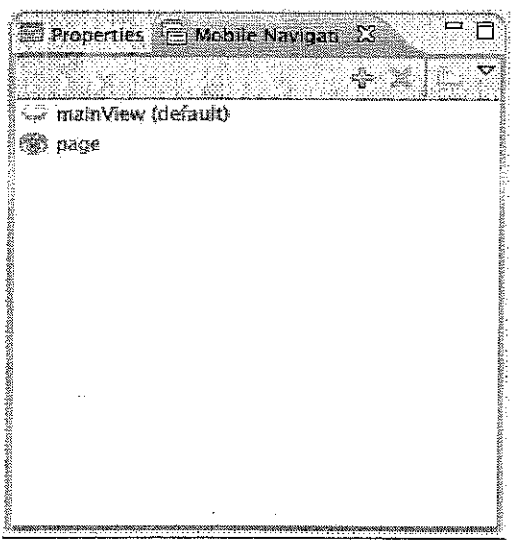
Deleted:



6. Next we'll add a second Page to our application by dragging Page widget from the jQuery Mobile Widgets section of the Pallet into the <body> of our Mysurance.html page, after the mainView Page as follows:



7. Open the Mobile Navigation tab and notice how you can alternate the contents of the design tab in the Rich Page Editor by selecting which is the active Mobile View – the view with the open eye icon is currently displayed in the design tab. Click the closed eyelid to make that Mobile View become the visible view in the editor. Click on the second one, i.e. "page", to switch to another page.



Deleted: <#>Remove the default "Mysurance" text that now sits below the ScrollableView <div> by deleting it from the source pane in the Rich Page Editor pane.¶
¶ [... [7]]

Deleted: dojox.mobile.ScrollableView

Deleted: another ScrollableView

Deleted: Dojo

Formatted: Font: Courier

Deleted: ScrollableView

Deleted:

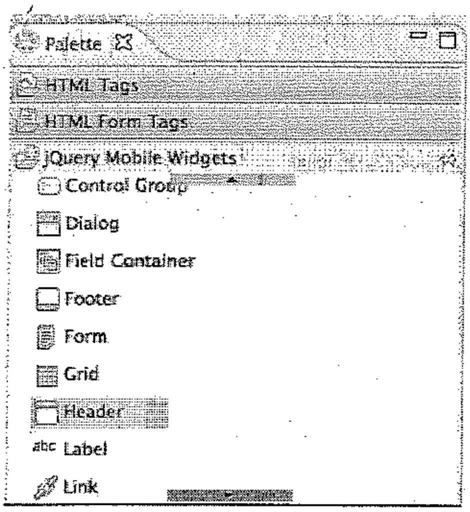
Formatted: Bullets and Numbering

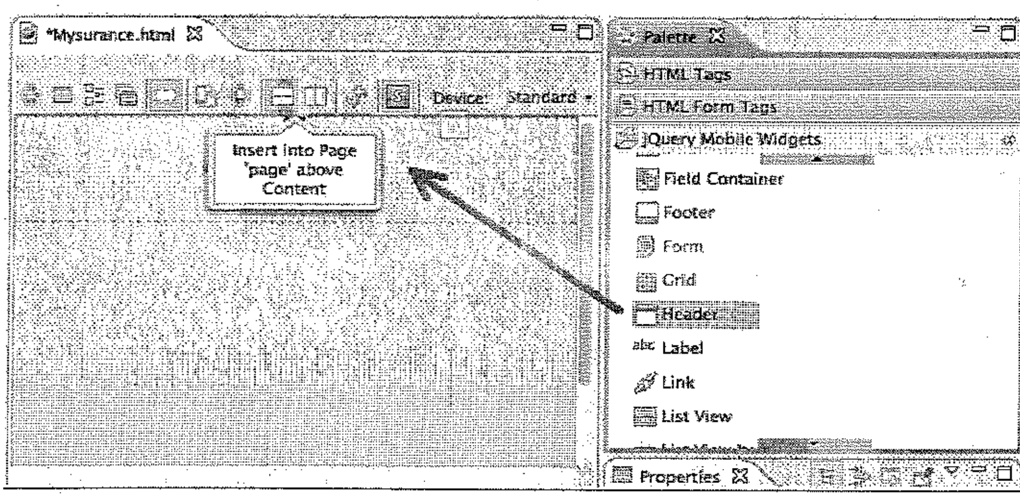
Deleted: After dragging and dropping, the 'Create a new Dojo Mobile Scrollable View' dialog will surface

Formatted: No bullets or numbering

__8. Add a header for the second page.

Drag the **Header** widget from the **jQuery Mobile Widgets** section of the palette into the design tab of the editor, in the `<div data-role="page" id="page">` section of the Mysurance.html file before the `<div data-role="content" style="padding: 15px">` section.





Formatted: Font: Courier, 11 pt

Formatted: Font: Courier, 11 pt

Formatted: No bullets or numbering

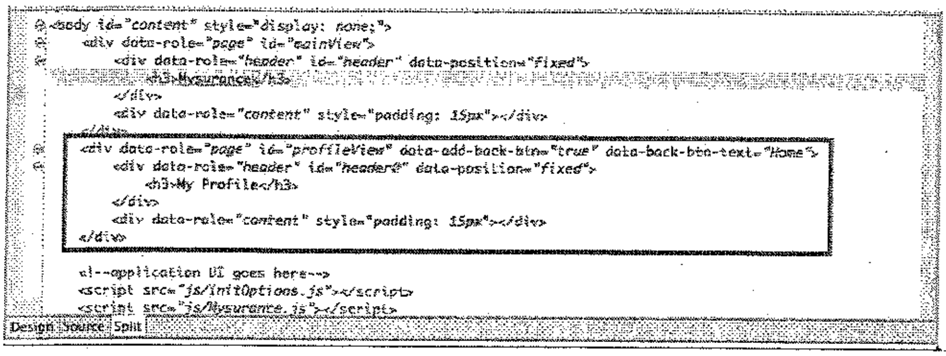
__9. Modify the HTML code of the page as follows.

__a. Change the `id` attribute of the page view from "page" to "profileView".

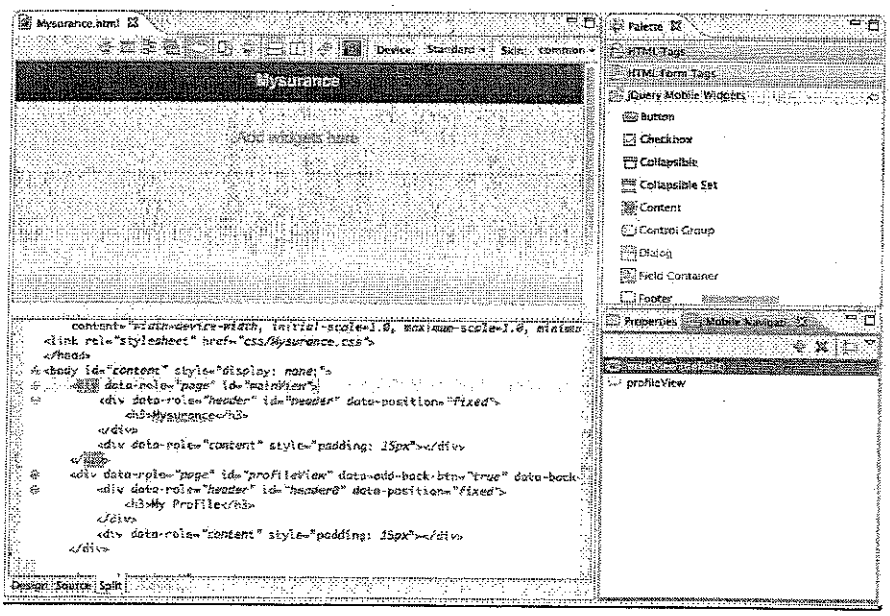
b. Change the heading label from "Header" to "My Profile".

c. Add the following code after the id attribute of the page.

```
data-add-back-btn="true" data-back-btn-text="Home"
```



__10. If you select the eyelid for the mainView, mainView is selected in the Mobile Navigation tab and rendered in the design tab:



Formatted: Font: Courier

Formatted

Formatted: Bullets and Numbering

Deleted: ¶

Formatted: Font: Courier

Formatted: Indent: Left: 1.06", No bullets or numbering

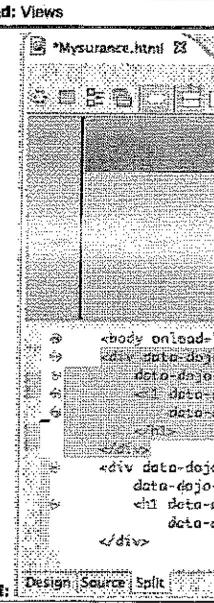
Formatted: Font: Courier, Not Highlight

Formatted: Indent: Left: 0", Hanging: 0.55", No bullets or numbering

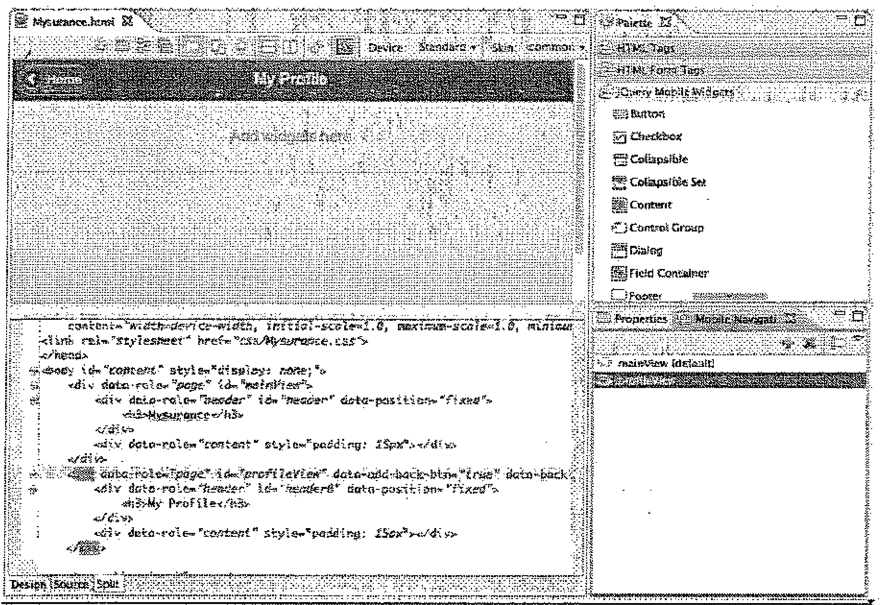
Formatted: Font: Monaco

Deleted: ¶
<#> Enter `profileView` as the Id.¶
<#>Select the **Include heading** checkbox to include a heading.¶
<#>Enter `My Profile` as the **Heading label.**¶
<#>Enter `Home` as the **Back button label** to add a Back button label named 'Home'.¶
<#>Select **mainView** from the **Back button target** drop-down box.¶
<#>Click **Finish** to generate the ScrollableView code for our Profile view.¶
<#>Open the Mobile Views tab and notice how you can alternate the contents of the design tab in the Rich Page Editor by selecting which ... [8]

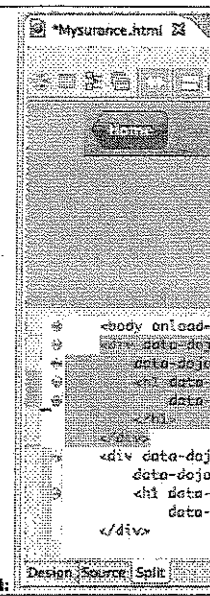
Deleted: Views

Deleted: 

If you select the eyelid for the profileView, profileView selected in the Mobile Navigation tab and rendered in the design tab.



Deleted: Views



Deleted:

Formatted: Bullets and Numbering

__11.__ Complete navigation between the views.

When we created the 2nd Page view (profileView), we also created a back button for navigation back to the Mysurance home screen (mainView). Now we'll add a toolbar button on the mainView which will send us to the profileView and complete our navigation.

Deleted: ScrollableView

Deleted: were given the option in the wizard to

__a. Make the **mainView** the active view in the Mobile Navigation tab, then go to the Palette. You can select the eyelid beside the mainView in the Mobile Navigation to select the mainView.

Deleted: Views

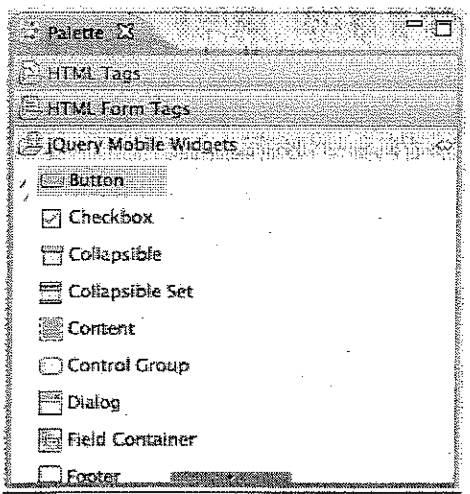
Deleted: View

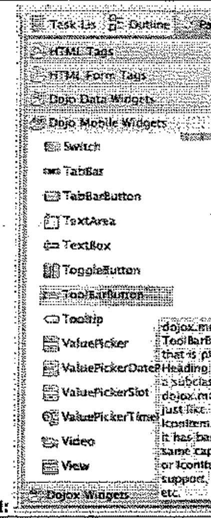
Deleted:

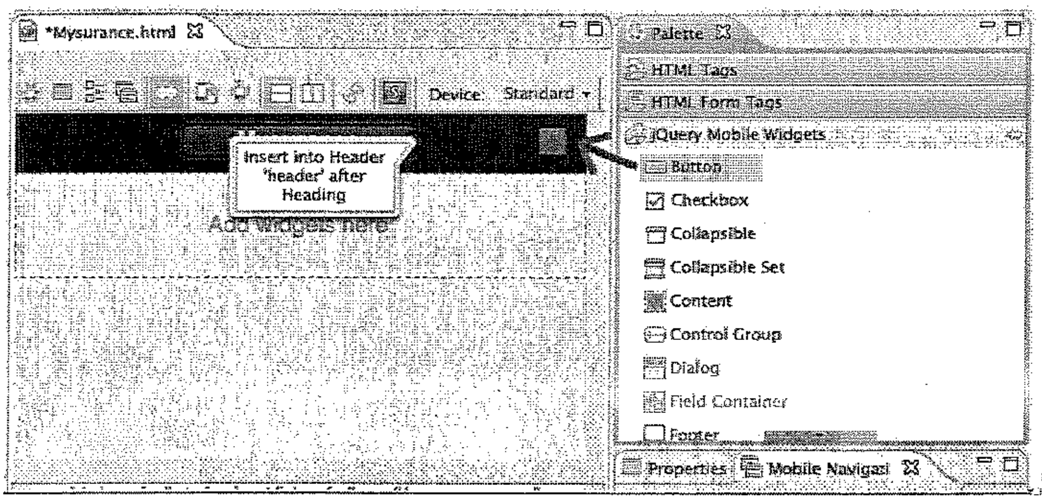
__b. Drag a **Button** onto the header portion of the mainView and **Insert into Header 'header' after Heading** (next to the 'Mysurance' header text)

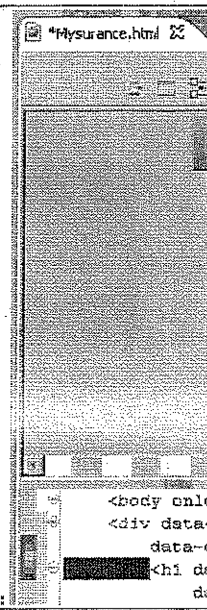
Deleted: ToolBar

Deleted: ing



Deleted: 



Deleted: 

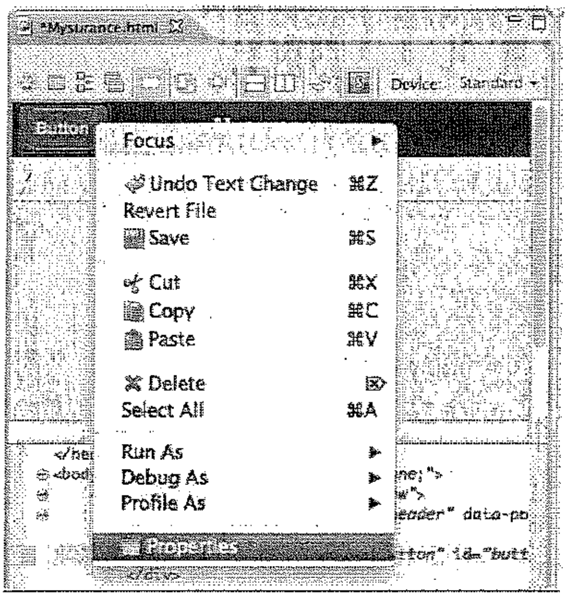
12. Link the new button to the profileView mobile view.

__a. Select the new button, which is currently called 'Button', right-click on it and then select "Properties".

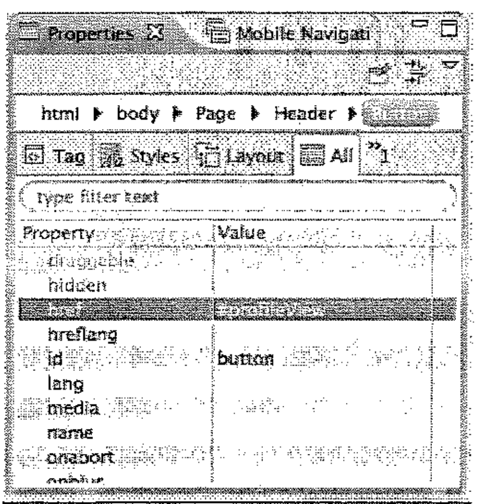
Formatted: Bullets and Numbering

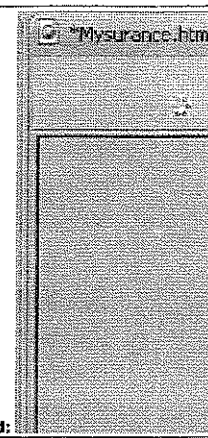
Deleted: Label'

Deleted: , and a hovering 'Link to Mobile View' icon should appear



__b. The Properties tab is opened. Look for property name "`href`" and set the value to "`#profileView`" (without quote).





Deleted:

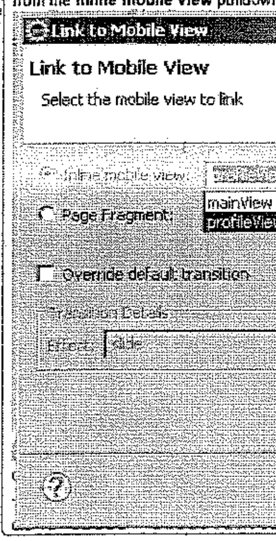
Deleted: Click the **Link to Mobile View** icon to open the Link to Mobile View dialog.

Formatted: Font: Courier

Formatted: Font: Courier

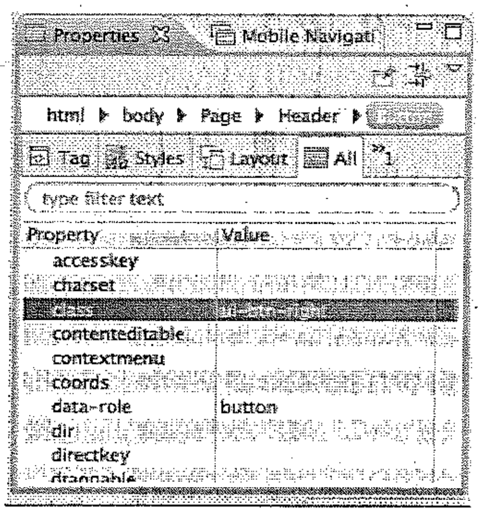
Formatted: Indent: Left: 0", Hanging: 0.55", No bullets or numbering

Deleted: <#>Select the **profileView** from the **Inline mobile view** pulldown.

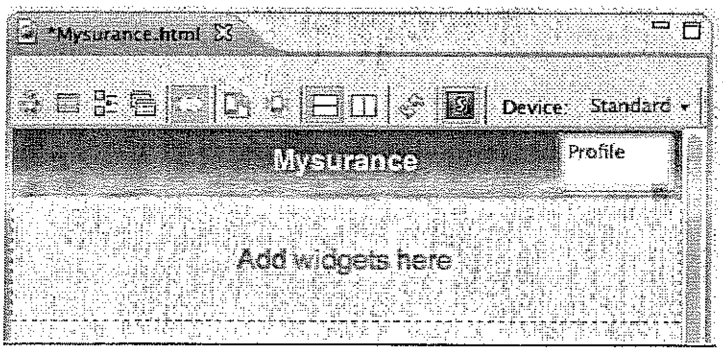


13. Next we'll change the button label from its default value of "Button" to "Profile", and add a CSS class name to relocate our button to the right hand side of the header.

   a. In the Properties tab, look for property name "`class`" and set the value to "`ui-btn-right`" (without quote).



   b. Double click the Button currently called 'Label'. It is now available to change the button label. Change the label from "Button" to "Profile".



Your button should be labeled 'Profile' and should be on the right hand side of the header now.

Deleted: <#>Click Finish to complete the link from this button on the mainView to the profileView. ¶

Deleted: Label

Deleted: some inline CSS

Formatted: Font: Courier

Formatted: Font: Courier

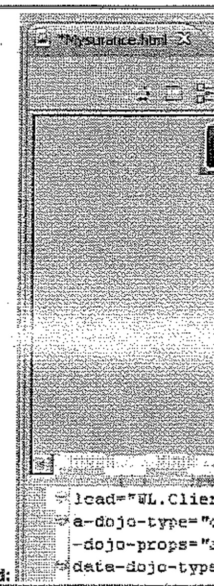
Formatted: Indent: Left: 0.55", No bullets or numbering

Deleted: Right click the ToolBarButton

Formatted: Bullets and Numbering

Deleted: and select **Properties** from the menu. You can double click on the Properties tab to maximize the view.

Deleted:

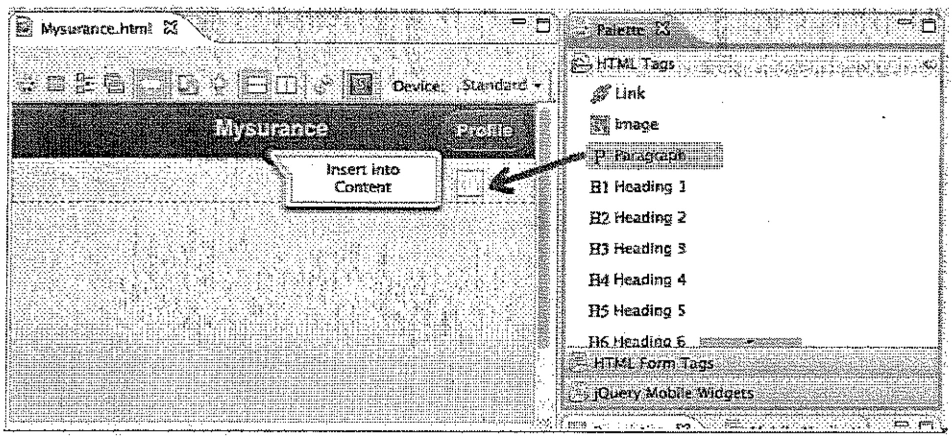


Formatted: Indent: Left: 0"

Deleted: <#>In the Dojo tab of the Properties view, update the Label with the new name for our button, 'Profile'. ¶ ... [9]

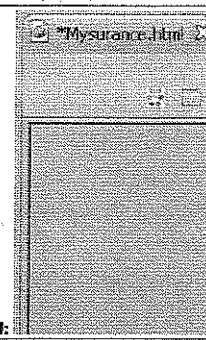
__14. Add text to the mainView

__a. Drag a **Paragraph** widget under HTML Tags from the palette into the mainView, inserting into the content.



__b. Paste the following text between the start and ending <p> tags for the paragraph in the source tab.

```
A sample application built on the <b>IBM Worklight Platform</b>, using
HTML5, CSS3, JavaScript and the Dojo Mobile framework.
```



Deleted:

Formatted: Bullets and Numbering

Deleted: in a Rounded Rectangle (dojox.mobile.RoundRect)

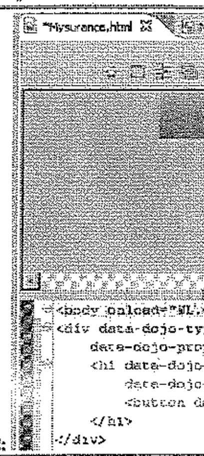
Formatted [... [10]]

Deleted: RoundRect

Deleted: ScrollableView

Deleted: below the Heading

Deleted: ¶



Deleted:

Deleted: Double-click the newly created RoundRect to bring it into view in the Source tab.

Deleted: div

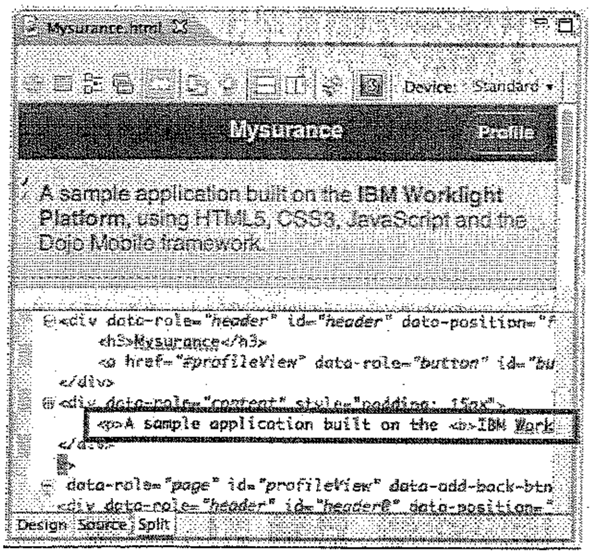
Deleted: RoundRect

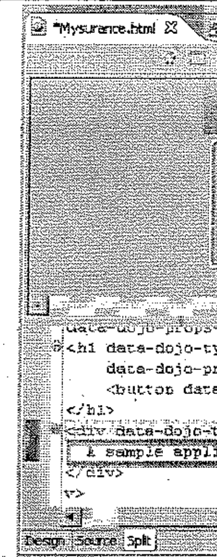
Deleted:

Formatted: Font: Courier

Formatted: Code - Paragraph style

Deleted: ¶
Use above or Snippet #3 from the accompanying lab1-snippets.txt file for [... [11]]





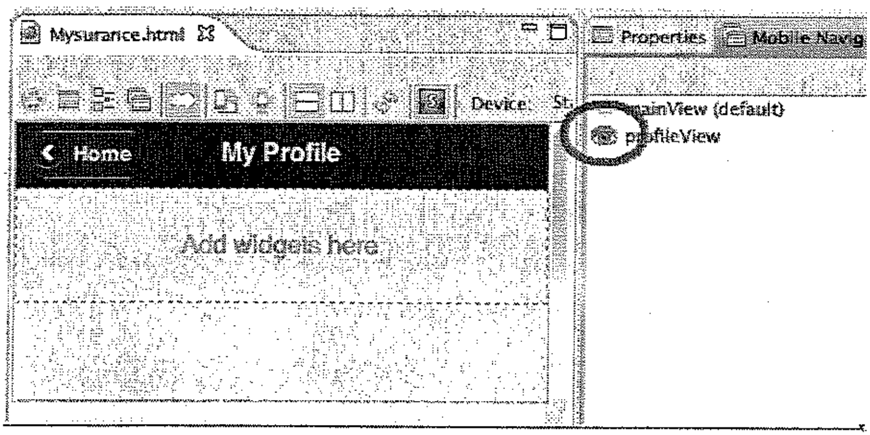
Deleted:

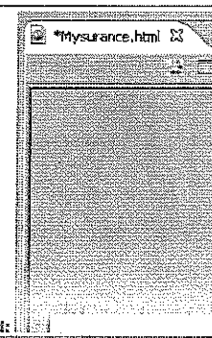
__15. Add a button to the profileView. This button will be used to launch a pop-up dialog box.

Formatted: Bullets and Numbering

Deleted:

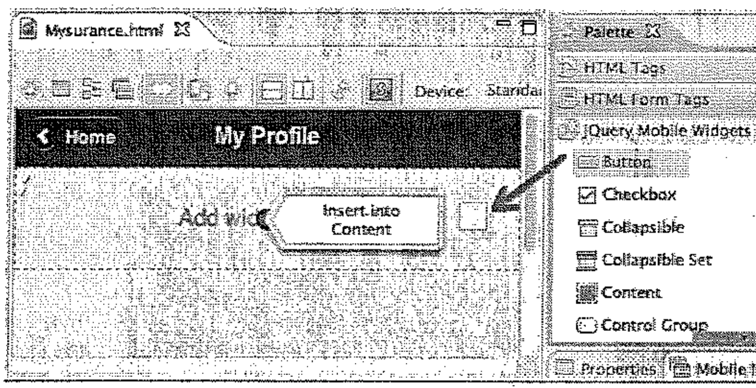
__a. In the Mobile View, click on the **eye** icon to change to the profileView page.

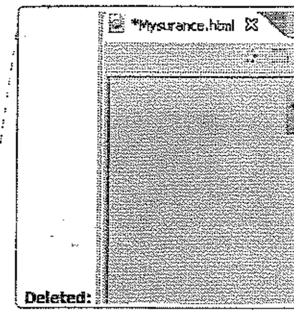




Deleted:

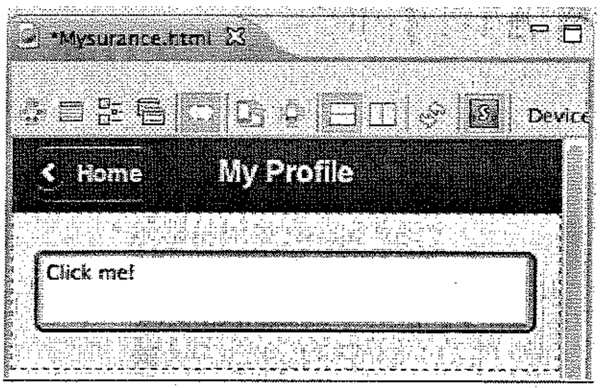
__b. Drag a **Button** from the palette and drop into the profileView below the Heading.





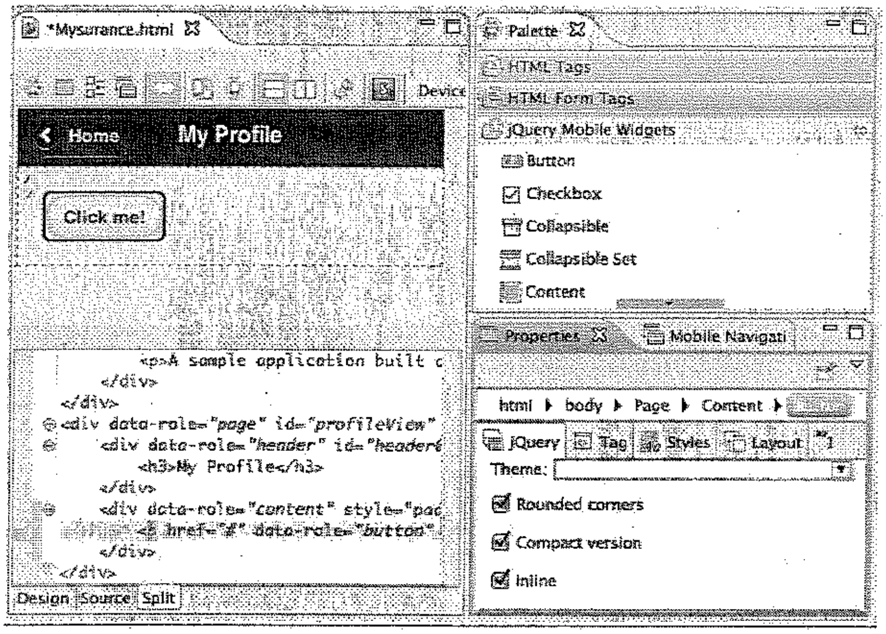
__c. Double click the Button, enter `Click me!` as the **label**.

Deleted: In the Button's properties page, change to the All tab. E

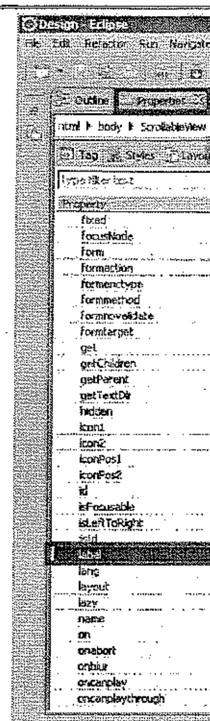


16. In the Button's properties page, change to the jQuery tab. Check the boxes for "Compact version" and "Inline".

Formatted: Outline numbered + Level: 1 + Numbering Style: 1, 2, 3, ... + Start at: 1 + Alignment: Left + Aligned at: 0" + Tab after: 0.55" + Indent at: 0.55"



Formatted: Indent: Left: 0", First line: 0"



Deleted:

Formatted: Bullets and Numbering

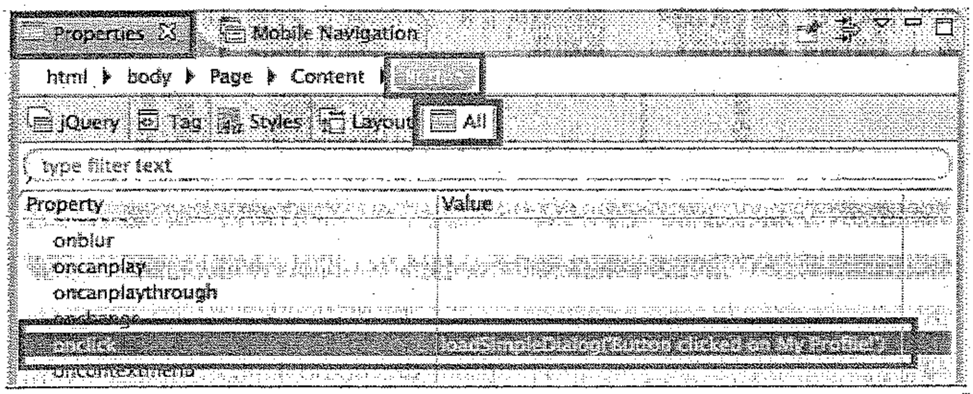
__17. Prepare the button to launch a dialog when clicked by adding the function to the onclick property (we'll implement this function in JavaScript in a few steps).

__a. In the same **properties** view – **All** tab, enter "`onclick`" in the **filter** field to narrow down the property list.

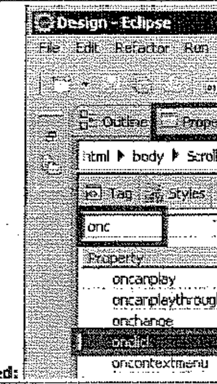
__b. In the **onclick** property, add the following function as the **value**.

```
loadSimpleDialog('Button clicked on My Profile!')
```

__c. Minimize the properties view and go back to the Mysurance.html editor.



Deleted:



Formatted: Bullets and Numbering

18. Add a Container (<div>) to the mainView.

Formatted: Font: Courier

This step is primarily cosmetic, but also accounts for some spacing inconsistencies in different browsers and environments. It is a very common HTML/CSS practice. It places <div> tags in our views that will be used by CSS for controlling the vertical positioning of our page content. It will provide visual separation from the Header and help account for spacing inconsistencies in different browsers and environments.

Formatted: Font: Courier

Deleted: (only RoundRect and Button, so far)

Deleted:

Deleted: Views

a. Use the **Mobile Navigation** tab to bring the **mainView** back to the foreground of the design tab.

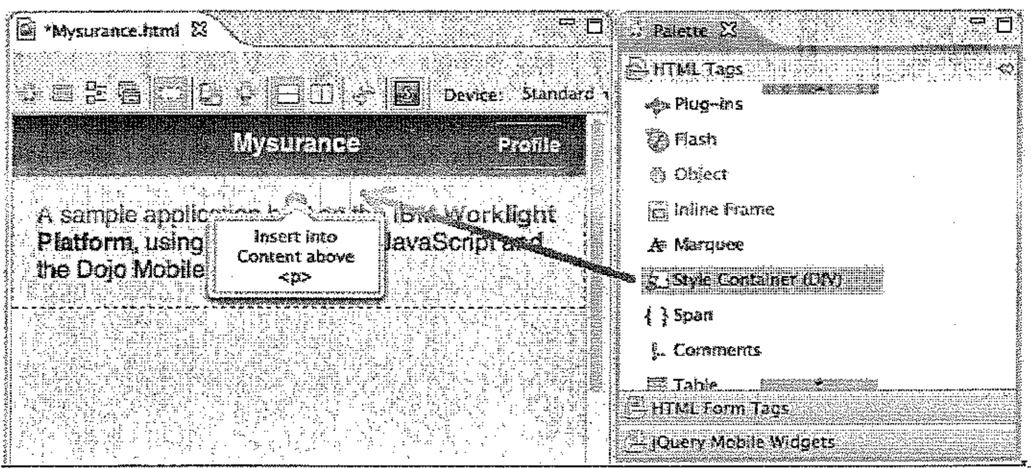
b. Add a Container (<div>) to the mainView by dragging the **Style Container** widget from the palette and inserting into the Content before the <p> tag.

Formatted: Font: Courier

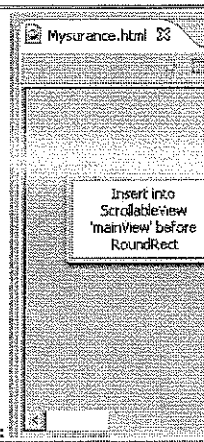
Deleted: ScrollableView

Deleted: RoundRect

Formatted: Font: Courier



Deleted:



Formatted: Bullets and Numbering

19. Add a class name to the Container, so that we can apply CSS for margins.

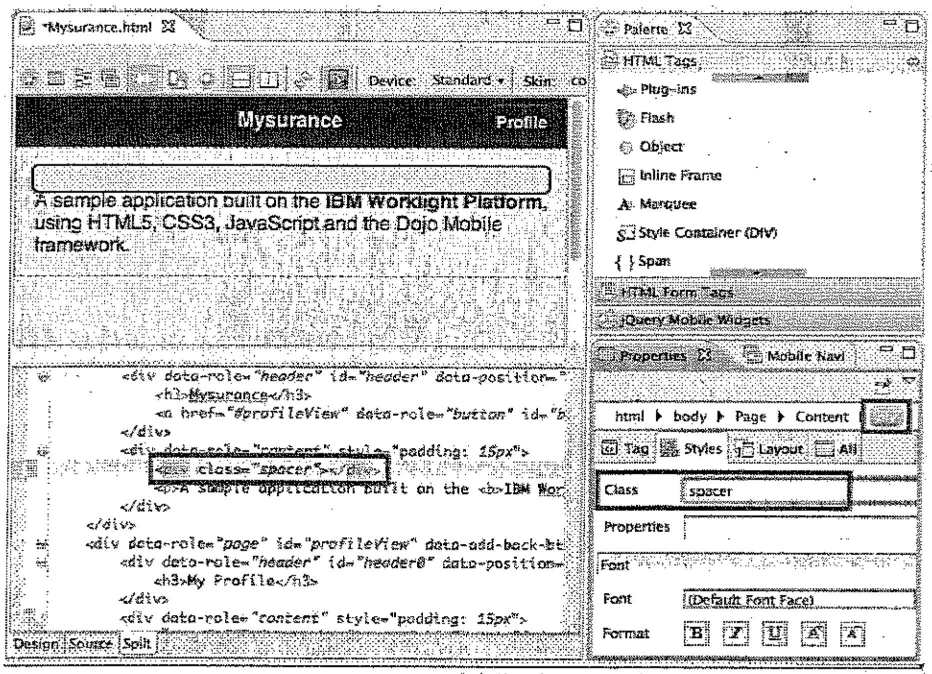
__a. Select the **Container** <div> line in the source pane.

Deleted: dojox.mobile.

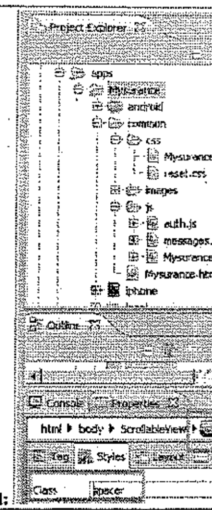
__b. Switch to the **Properties** view and ensure that the Container element is displayed.

__c. Switch to the **Styles** tab within the Properties view.

__d. Enter `spacer` as the value for the **Class** property. The CSS file will have a reference for the class named "spacer" which this container will inherit.



Deleted:



__20. Add a Container (`<div>`) to the profileView.

Formatted: Font: Courier

Formatted: Bullets and Numbering

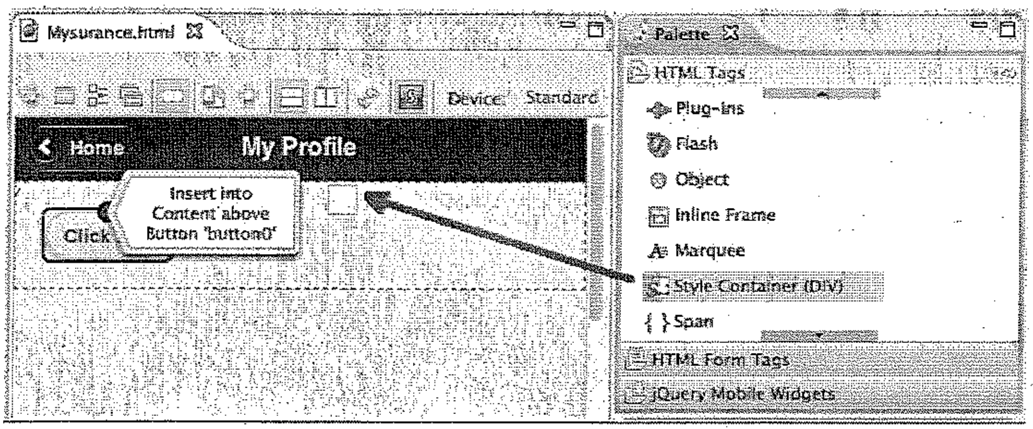
Deleted: Views

__a. Use the **Mobile Navigation** tab to bring the **profileView** back to the foreground of the design tab.

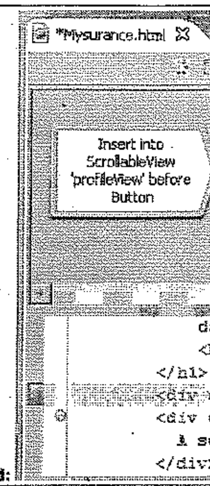
__b. Add a Container (`<div>`) to the profileView by dragging the **Style Container** widget from the palette and inserting into the Content before the Button.

Formatted: Font: Courier

Deleted: ScrollableView



Deleted:

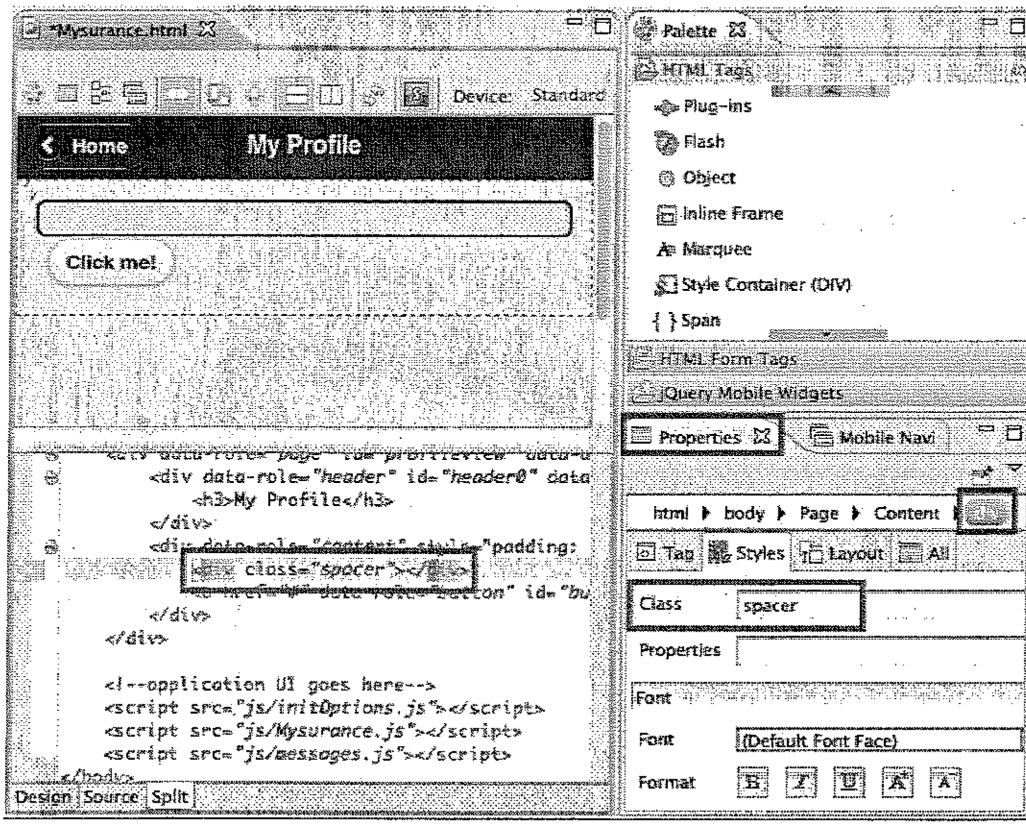


Formatted: Bullets and Numbering

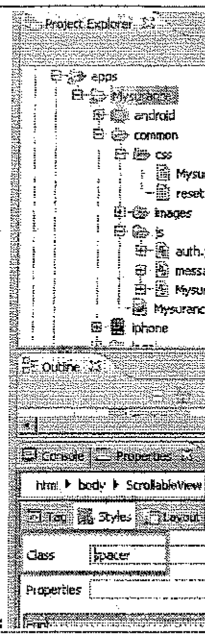
Deleted: dojox.mobile.

__21.__ Add a class name to the Container, so that we can apply CSS for margins.

- __a. Select the **Container** <div> line in the source pane.
- __b. Switch to the **Properties** view and ensure that the Container element is displayed.
- __c. Switch to the **Styles** tab within the Properties view.
- __d. Enter `spacer` as the value for the **Class** property. The CSS file will have a reference for the class named "spacer" which this container will inherit.



Deleted:



__22. Add CSS to Mysurance.css to provide a "branded" look the Mysurance application, overriding defaults and giving a more consistent experience across different devices.

Formatted: Bullets and Numbering

__a. In the Project Explorer view, find and open **/Mysurance/apps/Mysurance/common/css/Mysurance.css** in the editor, and paste the following CSS text to the bottom of the file:

Deleted: ¶

Formatted: No bullets or numbering

```
.spacer {
      margin-top: 100px;
}

.ui-page {
      background: -webkit-gradient(linear, left top, left bottom,
from(#096093), to(#ffffff)) !important;
}

.ui-content p {
```

Formatted: Font: Courier

Formatted: Indent: Left: 0.55", No bullets or numbering

Formatted: Font: Courier, 11 pt

Formatted: Font: Courier

Formatted: Font: Courier

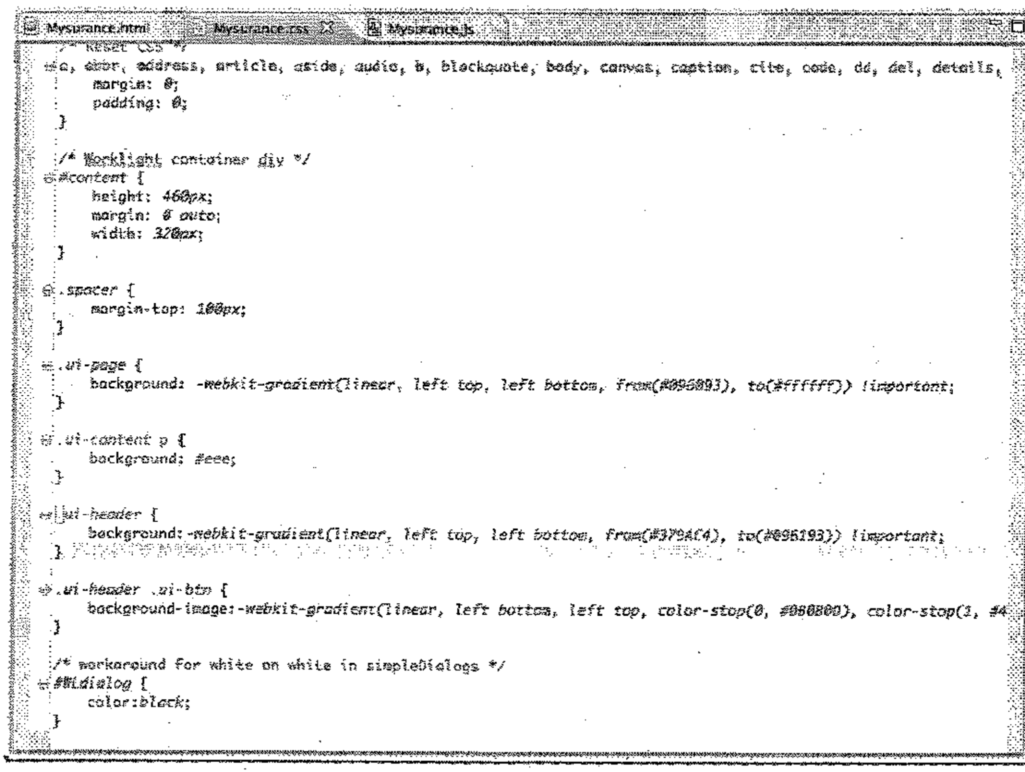
Formatted: Font: Courier, 11 pt

```
    background: #eee;
}

.ui-header {
    background:-webkit-gradient(linear, left top, left bottom,
from(#379AC4), to(#096193)) !important;
}

.ui-header .ui-btn {
    background-image:-webkit-gradient(linear, left bottom, left top,
color-stop(0, #0B0B0D), color-stop(1, #434949)) !important;
}

/* workaround for white on white in simpleDialogs */
#WLdialog {
    color:black;
}
```



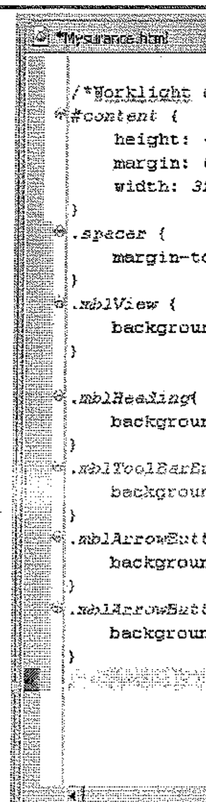
23. Add the JavaScript function to implement our Simple Dialog pop-up.

Formatted: Font: Courier

Deleted: .spacer {¶
 margin-top: 100px;¶
}¶
.mblView {¶
 background: -webkit-gradient(linear, left top, left bottom, from(#096093), to(#ffffff)) !important;¶
}¶
¶
.mblHeading{¶
 background:-webkit-gradient(linear, left top, left bottom, from(#379AC4), to(#096193)) !important;¶ ... [12]

Formatted: Normal, Indent: Left: 0"

Deleted: ¶
¶
Use above or Snippet #4 from the accompanying lab1-snippets.txt file for cut & paste.¶
¶

Deleted: 

Formatted: Bullets and Numbering

The final step before previewing is to add the function that the "Click Me!" button on the profileView will invoke to generate a popup message on the device.

To display this message, we will use a Worklight Common Control, **WL.SimpleDialog**. Worklight provides a set of Common Controls as part of the JavaScript library that offers simplified and consistent on-device options for common activities and visual elements. It includes things like busy indicators, modal popup windows, tab bars and options menus. Some of these are also available in JavaScript, some in Dojo and other JavaScript libraries, and some only in native SDKs, with differences in support on various devices. Providing as WL.* Common Controls improves the **write-once, run-anywhere** development experience.

__a. In the Project Explorer view, open the Mysurance.js file by navigating to **Mysurance/apps/Mysurance/common/js/Mysurance.js**.

__b. Paste the following function text into the bottom of Mysurance.js.

```
function loadSimpleDialog(dialogText) {
        WL.SimpleDialog.show('Hello!', dialogText, [{
            text: 'Dismiss',
            handler: function()
                {WL.Logger.debug("button pressed"); }
        }]);
}
```

This function accepts a string from the caller (*dialogText*), and invokes the Worklight Common Control **WL.SimpleDialog.show()**, passing a title for the dialog, the text to display in the dialog, and details about the dismiss button, which offers the ability to call a function as the dialog is dismissed.

__24.__ Save all open files and return to Mysurance.html.

On certain combinations of platform and Rich Page Editor browser selection (browser buttons on the menu bar of the editor pane or Window > Preferences > Web > Rich Page Editor), you will see the CSS changes reflected immediately in the Design tab. Otherwise they will always be available during Preview in the Mobile Browser Simulator, or when the application is launched on an emulator or an actual device.

Formatted: Font: Courier

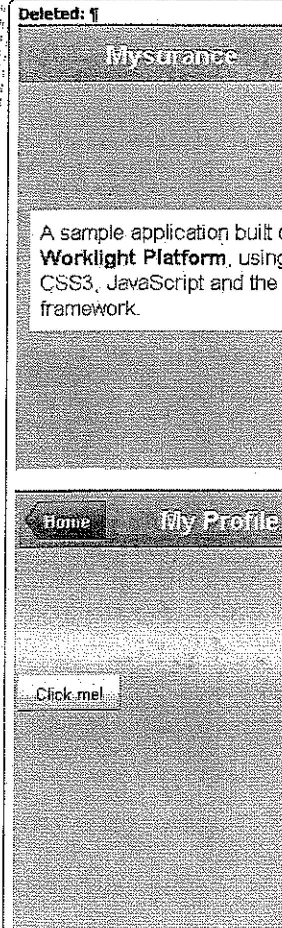
Deleted: Use above or Snippet #5 from the accompanying lab1-snippets.txt file for cut & paste.¶

Formatted: Bullets and Numbering

Formatted: Indent: First line: 0"

Deleted: ¶
On

Deleted: ¶



¶
¶
¶
¶

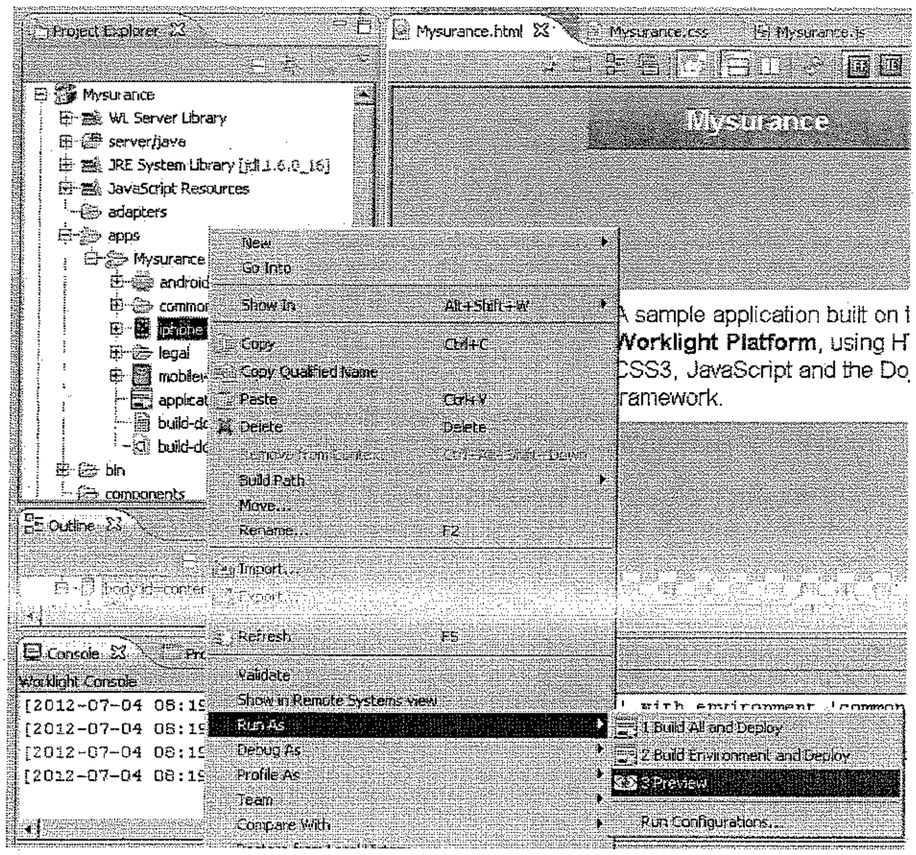
1.6 Testing Android and iPhone versions in the Mobile Browser Simulator

In this section we'll see the strength of the new Mobile Browser Simulator for development time rendering in an integrated, device-specific container.

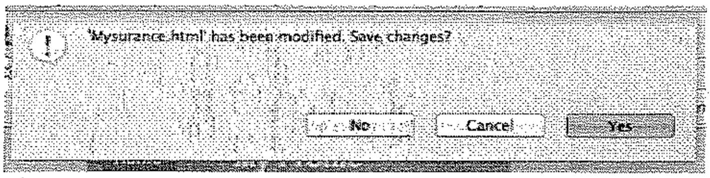
__1. Preview the iphone environment in the Mobile Browser Simulator

__a. In the Project Explorer view, expand the **Mysurance** project > expand **apps** folder > expand **Mysurance** folder and right-click **iphone** environment > **Run As** > **Preview**.

__b. This will show the application and how the navigation works on the iphone environment.

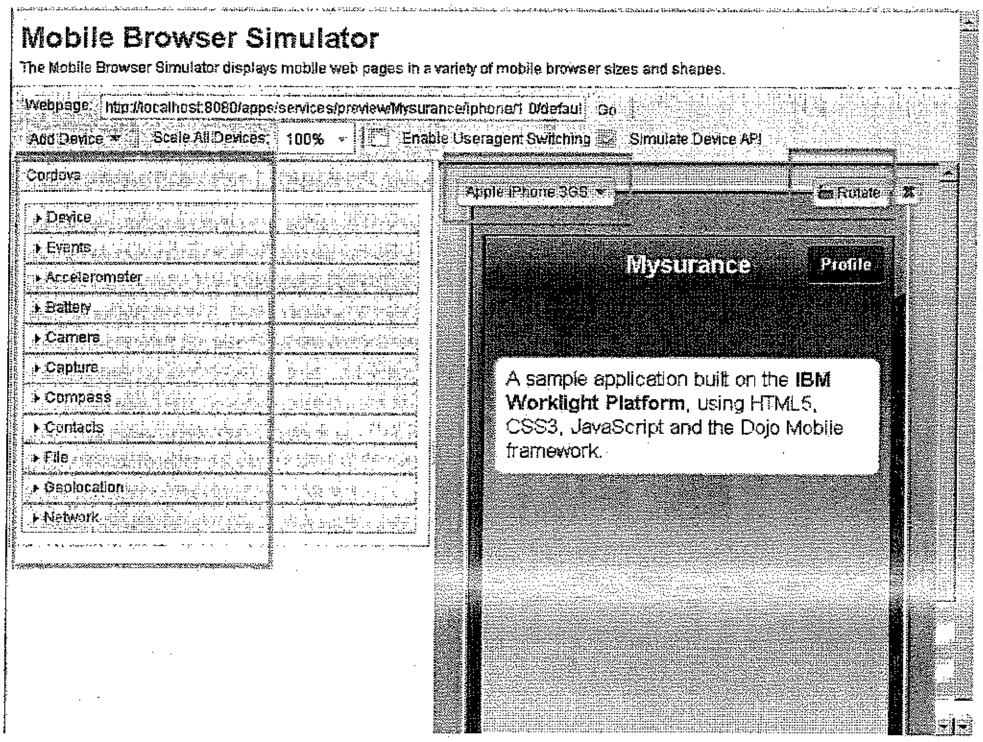


The studio will force a rebuild and prompt to save any open files if necessary. Select **Yes** if prompted:

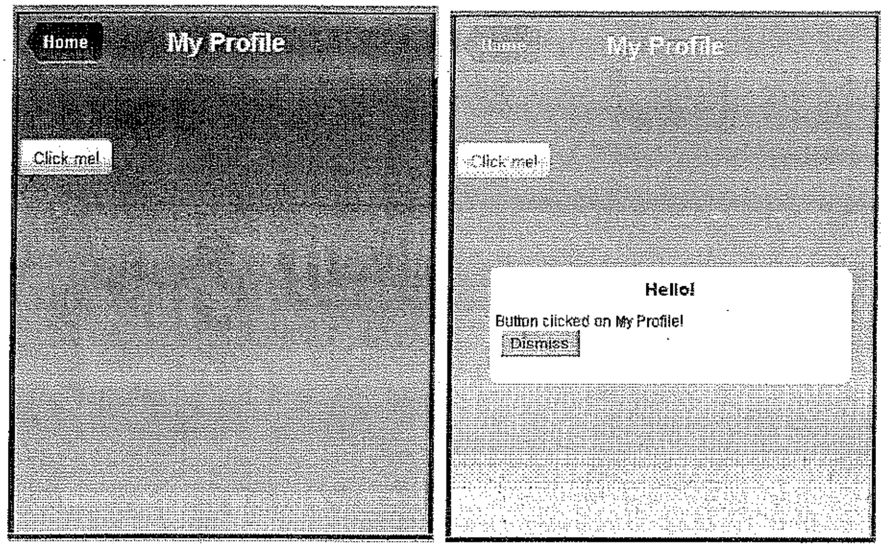


Note: As the simulator opens, you may need to accept the prompts to allow Java to execute on this page.

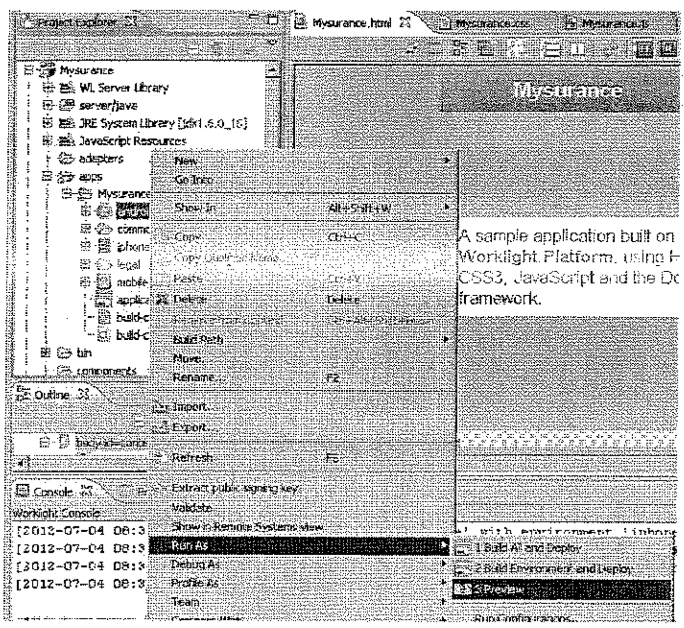
Notice the Cordova settings in the left hand pane, which allow you to test changes to various on-device settings within the Mobile Browser Simulator. Also the Device selection menu and orientation options at the top of the Simulated device window.



Navigate between the Mysurance and My Profile 'pages' to test our navigation, and try the **Click Me!** Button on the profile page to bring up our Simple Dialog.

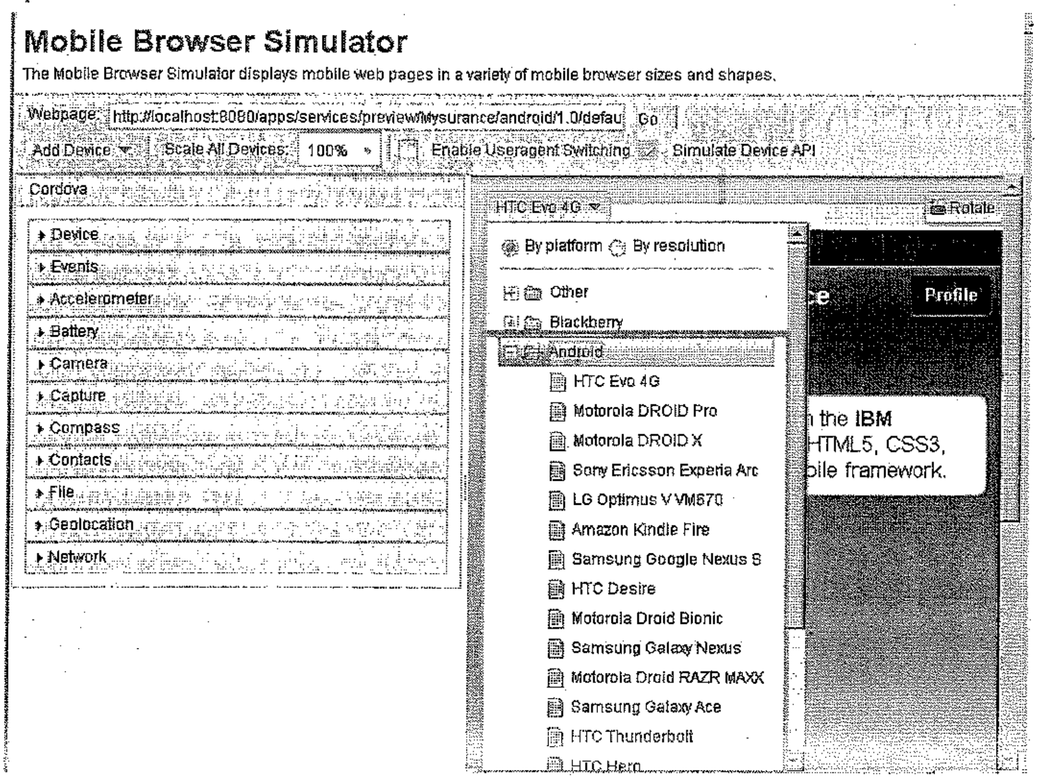


__2. Repeat **preview** for Android by selecting the android application node in the Project Explorer (**Mysurance/apps/Mysurance/android**), right click and **Run As > Preview.**



Use the simulator to review the application function on Android. Navigation, look and feel should be very similar to the iphone, due to our custom CSS.

Expand the pulldown on the current device selection (HTC Evo 4G), then expand the Android section and notice the large number of Android device options for simulation.



Appendix - References

IBM Worklight Developer Edition

http://www.ibm.com/developerworks/mobile/worklight.html

Getting Started

http://www-01.ibm.com/software/mobile-solutions/worklight/library/

jQuery

http://www.jquery.com/

jQuery Mobile

http://jquerymobile.com/

Documents

http://pic.dhe.ibm.com/infocenter/wrklight/v5r0m5/index.jsp

https://www.ibm.com/developerworks/mobile/worklight/getting-started/current-documentation.html

http://api.jquery.com/

http://jquerymobile.com/demos/1.2.0/

Deleted: ¶

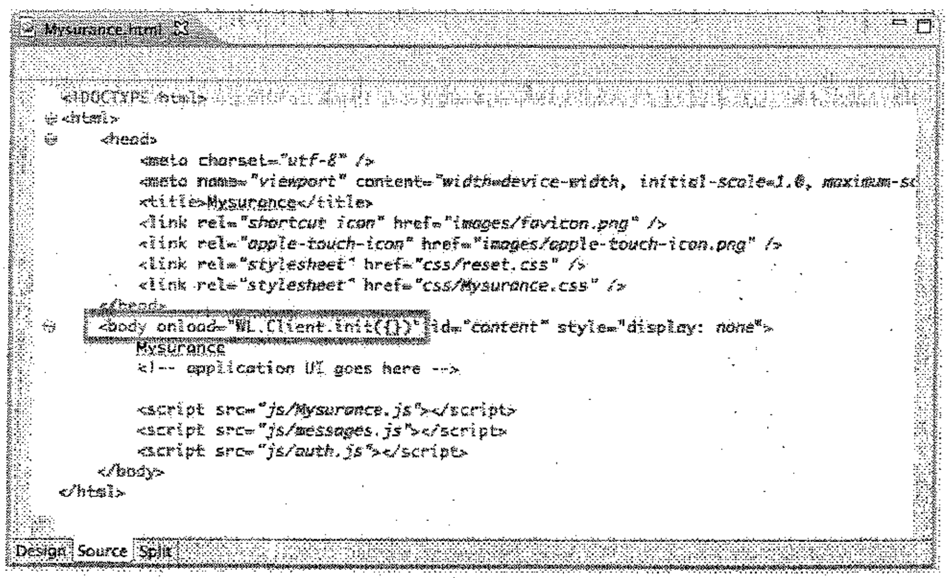
Forum

https://www.ibm.com/developerworks/forums/forum.jspa?forumID=2830

Deleted: ¶
¶
¶

Page 12: [1] Deleted **EricB Chan** **1/7/2013 5:10:00 PM**

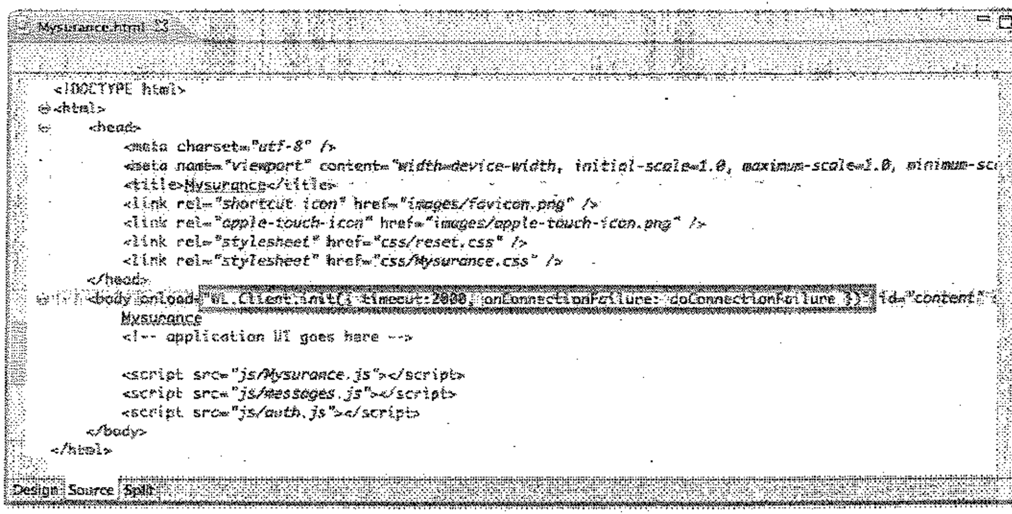
During the runtime of an application, the main HTML document cannot be replaced by another HTML document. The Worklight Client framework initialization is bound to the body during the **onload** event occurrence. For possible init options, see the Worklight Developer Reference Guide.



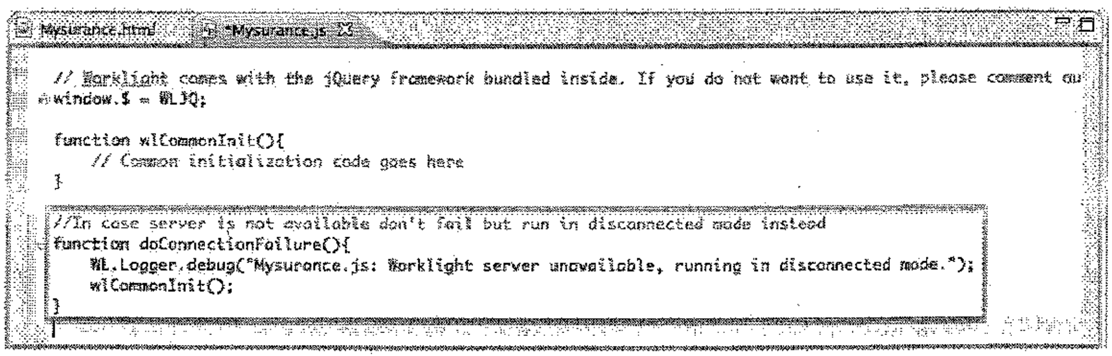
Page 12: [2] Deleted **EricB Chan** **1/7/2013 3:26:00 PM**

In the init method, enter "`timeout:2000, onConnectionFailure: doConnectionFailure`" (without the quote) inside the onload method of the <body> tag.

Page 13: [3] Deleted **EricB Chan** **1/7/2013 3:30:00 PM**



Expand the **common** then **js** folder, open the **Mysurance.js** file. This is the application's main JavaScript file. It contains the wlCommonInit() function that will be invoked automatically once the Worklight framework initialization completes. You can add your application's initialization code here.



You will add the callback function that was specified in init method earlier by adding in the following portion as shown in the image above.

```
// in case server is not available don't fail but run in disconnected mode
instead

function doConnectionFailure(){
        WL.Logger.debug("Mysurance.js: Worklight server unavailable,
running in disconnected mode.");
        wlCommonInit();
}
```

Use above or Snippet #2 from the accompanying lab1-snippets.txt file for cut & paste.

Save the Mysurance.js file

Page 25: [4] Deleted EricB Chan 1/7/2013 2:36:00 PM

"Pages" in a mobile application are built on views. For this application we are using dojox.mobile.ScrollableViews, which are dojo view widgets that manage a view pane. Ensure that the Mysurance.html file is the active file in the editor pane, and that the design or split editor tab is visible.

Page 26: [5] Formatted EricB Chan 1/7/2013 5:35:00 PM

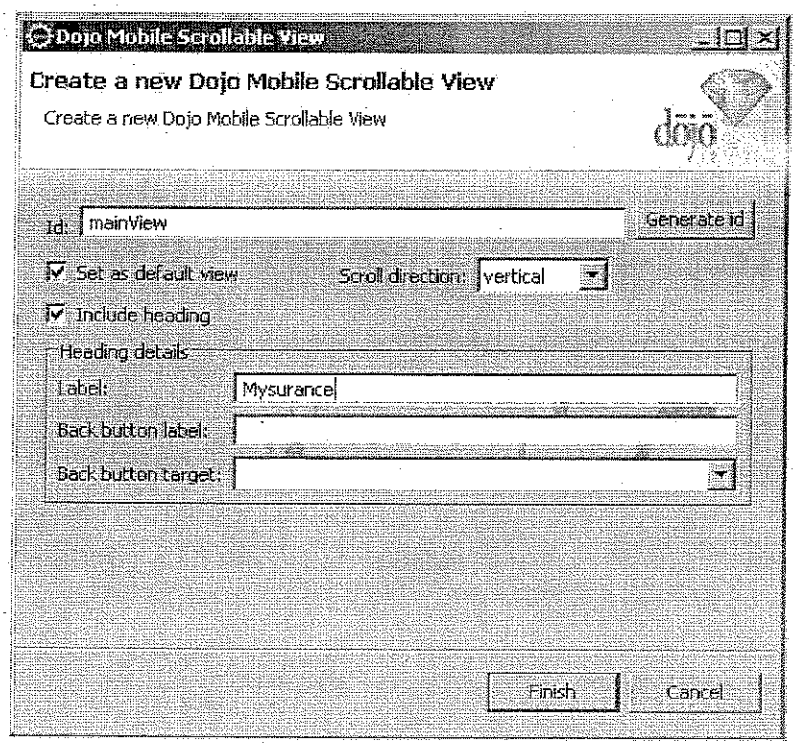
Step List, Outline numbered + Level: 2 + Numbering Style: a, b, c, … + Start at: 1 + Alignment: Left + Aligned at: 0.55" + Tab after: 0.55" + Indent at: 1.06"

Page 26: [6] Deleted EricB Chan 1/7/2013 5:32:00 PM

Select the **Include heading** checkbox.

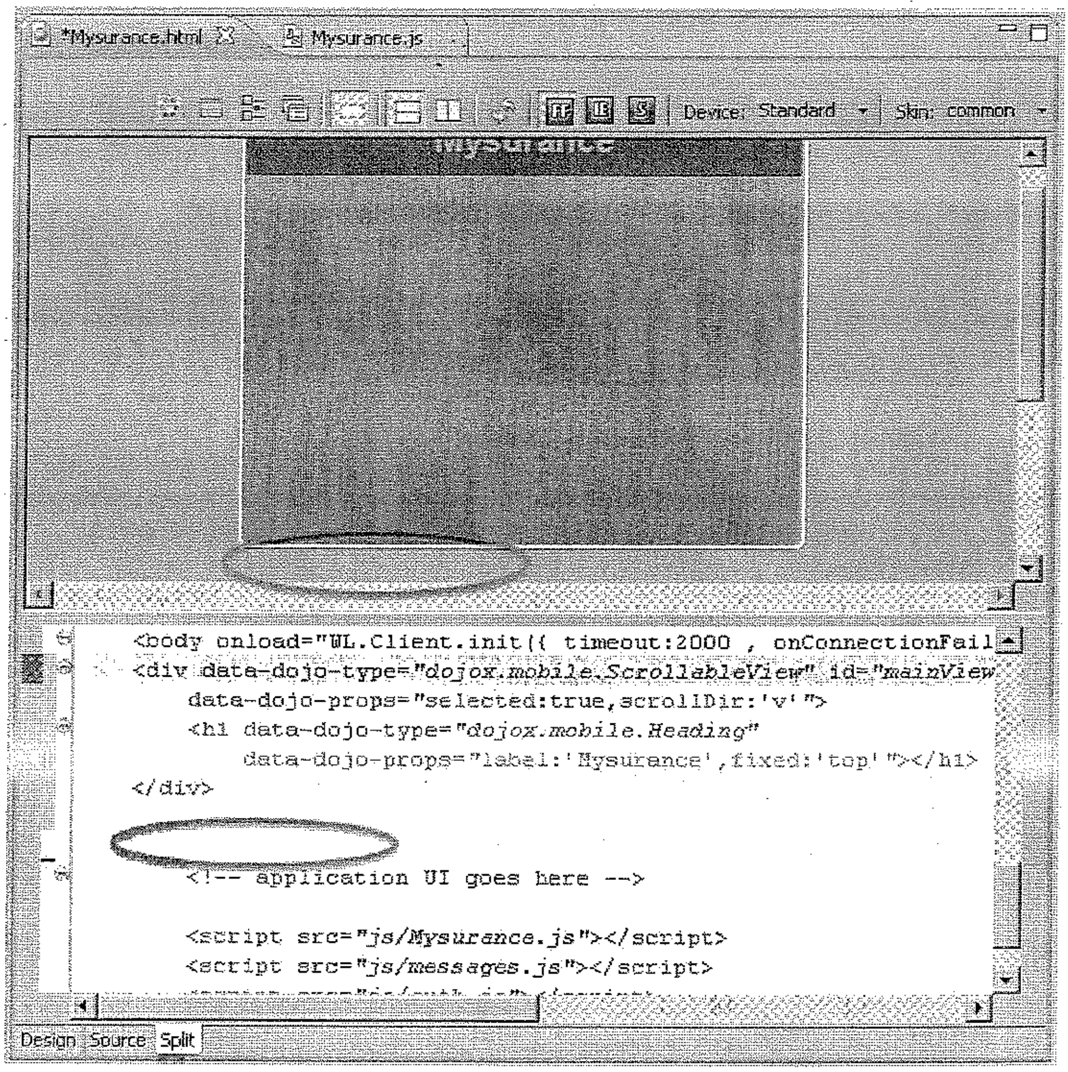
Enter `Mysurance` as the heading **label**.

Click **Finish** to complete the ScrollableView. If you have the Split tab open in the editor, you can see both the visual content and the source being added to your application



Page 27: [7] Deleted EricB Chan 1/7/2013 5:34:00 PM

Remove the default "Mysurance" text that now sits below the ScrollableView <div> by deleting it from the source pane in the Rich Page Editor pane.



Page 29: [8] Deleted EricB Chan 1/7/2013 6:13:00 PM

Enter `profileView` as the **Id.**

Select the **Include heading** checkbox to include a heading.

Enter `My Profile` as the **Heading label.**

Enter `Home` as the **Back button label** to add a Back button label named 'Home'.

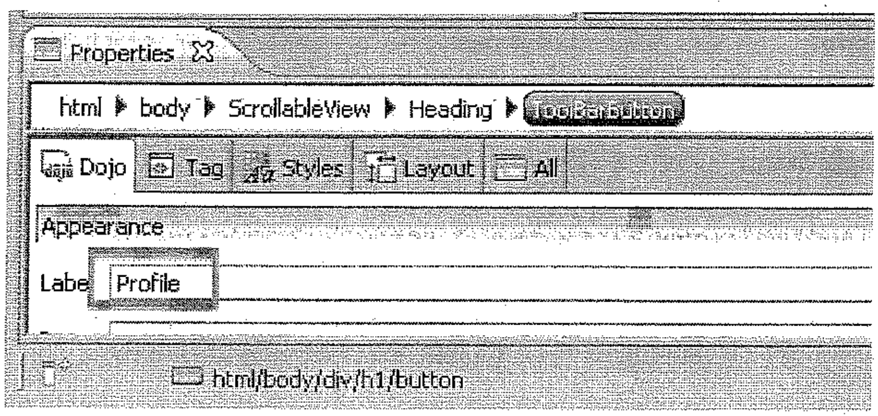
Select **mainView** from the **Back button target** drop-down box.

Click **Finish** to generate the ScrollableView code for our Profile view.

Open the Mobile Views tab and notice how you can alternate the contents of the design tab in the Rich Page Editor by selecting which is the active Mobile View – the view with the open eye icon is currently displayed in the design tab. Click the closed eyelid to make that Mobile View become the visible view in the editor.

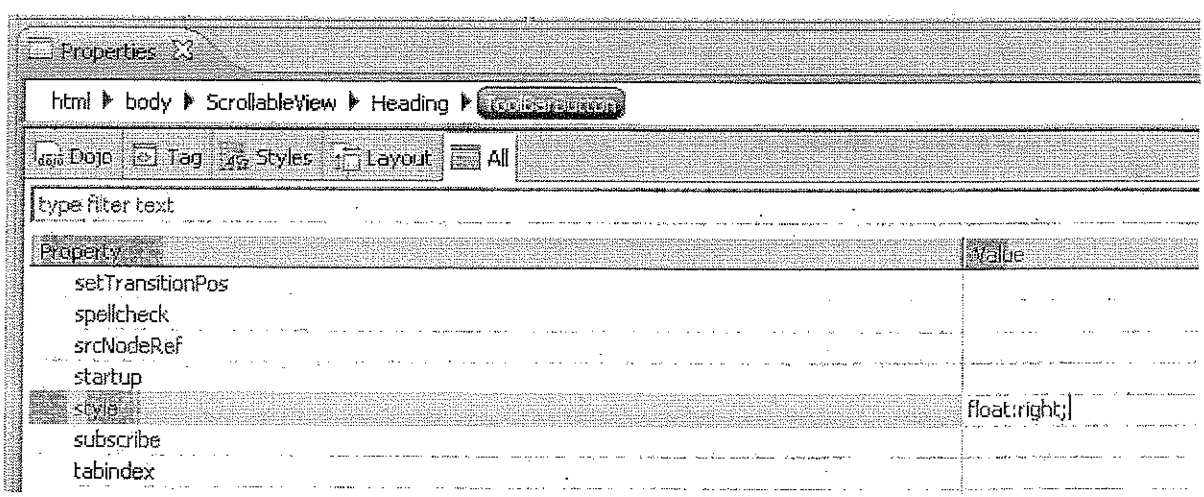
Page 33: [9] Deleted **EricB Chan** **1/7/2013 10:55:00 PM**

In the Dojo tab of the Properties view, update the Label with the new name for our button, '**Profile**'.



Switch to the **All** tab within the Properties view and scroll down to the **style** property.

Enter `float:right;` as the **Value** for the **style** field. Press **Enter** to make sure the value is accepted. This will force the button to appear on the right hand side of the header.



Page 34: [10] Formatted EricB Chan 1/7/2013 11:25:00 PM

Outline numbered + Level: 2 + Numbering Style: a, b, c, … + Start at: 1 + Alignment: Left + Aligned at: 0.55" + Tab after: 0.55" + Indent at: 1.06"

Page 34: [11] Deleted EricB Chan 1/7/2013 11:29:00 PM

```
Use above or Snippet #3 from the accompanying lab1-
snippets.txt file for cut & paste.
```

Page 42: [12] Deleted EricB Chan 1/8/2013 12:16:00 AM

```
.spacer {

	margin-top: 100px;

}

.mblView {

	background: -webkit-gradient(linear, left top, left bottom, from(#096093), to(#ffffff)) !important;

}


.mblHeading{

	background:-webkit-gradient(linear, left top, left bottom, from(#379AC4), to(#096193)) !important;

}

.mblToolBarButtonBody{

	background-image:-webkit-gradient(linear, left bottom, left top, color-stop(0, #0B0B0D), color-stop(1, #434949)) !important;

}

.mblArrowButtonBody{

	background-image:-webkit-gradient(linear, left bottom, left top, color-stop(0, #0B0B0D), color-stop(1, #434949)) !important;
```

```
}

.mblArrowButtonHead{

        background-image:-webkit-gradient(linear, left bottom, left top, color-stop(0,
#0B0B0D), color-stop(1, #434949)) !important;

}

/* workaround for white on white in simpleDialogs */

#WLdialog {

color:black;

}
```

IBM Inter-University
Programming Contest 2013

Chapter 6: DB2

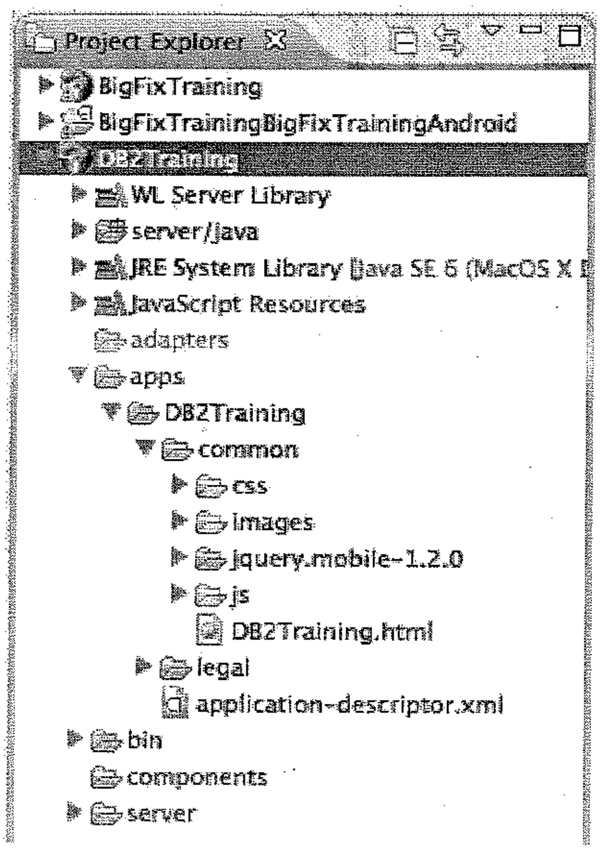
Objectives

In this exercise, we will learn:

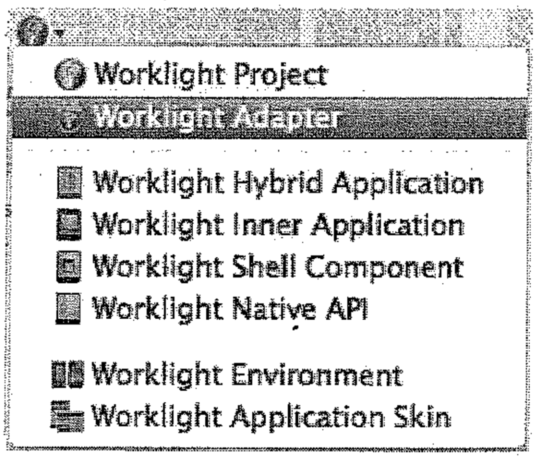
- Review response from DB2 server
- Perform database records request from Worklight to DB2 Server
- Parsing the result and integrate with mobile app

Exercises

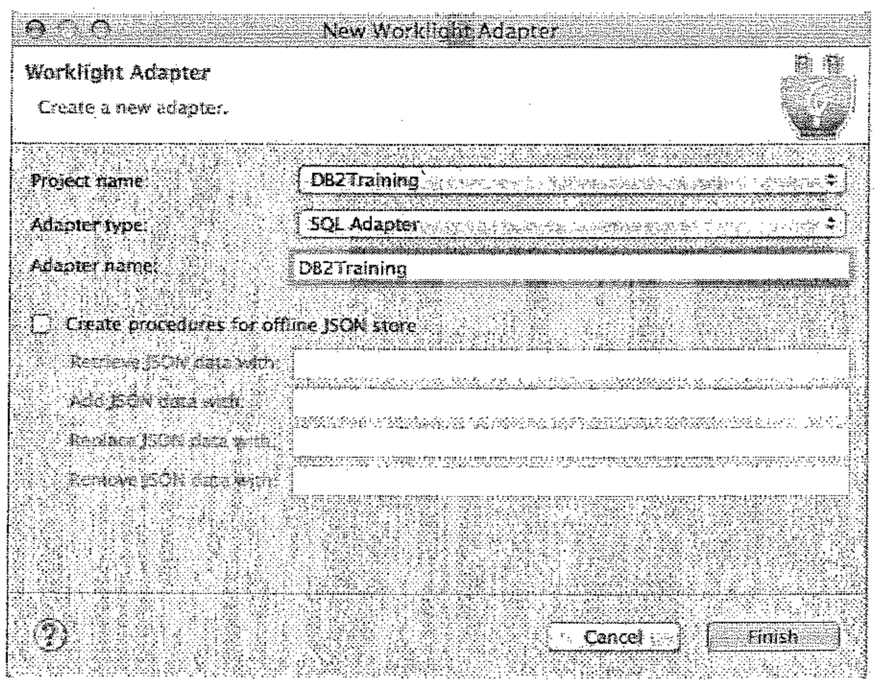
1. Open DB2Training as the active project to use.



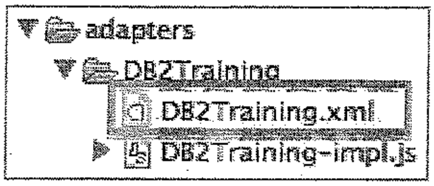
2. Let's make the Worklight enable to communicate with the BigFix server. First, create an HTTP Adapter in the Worklight Project. Find the Worklight button , click the down arrow. Then select "*Worklight Adapter*".



3. Choose the current project (i.e. *DB2Training*) in the field "Project name" and choose "*SQL Adapter*" for the Adapter Type. Enter a name for the adapter, here we will use "*DB2Adapter*".



4. Configure the adapter with the XML file. Open "*DB2Training.xml*".



5. Define the connectivity settings as follows:

   - Driver class: `com.ibm.db2.jcc.DB2Driver`
   - URL: `jdbc:db2://192.168.120.170:50000/TEST`
   - User: `db2admin`
   - Password: `password`

```
<dataSourceDefinition>
    <driverClass>com.ibm.db2.jcc.DB2Driver</driverClass>
    <url>jdbc:db2://192.168.120.170:50000/TEST</url>
    <user>db2admin</user>
    <password>password</password>
</dataSourceDefinition>
```

6. Declare adapter procedures. Remove all pre-defined procedures and define a new procedure called "*Get*".

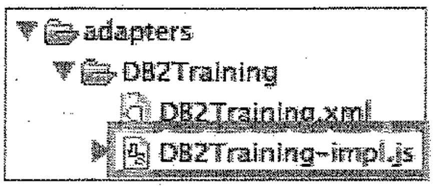
```
<wl:adapter name="DB2Training"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wl="http://www.worklight.com/integration"
    xmlns:sql="http://www.worklight.com/integration/sql">

    <displayName>DB2Training</displayName>
    <description>DB2Training</description>
    <connectivity>
        <connectionPolicy xsi:type="sql:SQLConnectionPolicy">
            <dataSourceDefinition>
                <driverClass>com.ibm.db2.jcc.DB2Driver</driverClass>
                <url>jdbc:db2://192.168.120.170:50000/TEST</url>
                <user>db2admin</user>
                <password>password</password>
            </dataSourceDefinition>
        </connectionPolicy>
        <loadConstraints maxConcurrentConnectionsPerNode="5" />
    </connectivity>

    <procedure name="Get"/>

</wl:adapter>
```

7. Implement procedures in the JavaScript file. Open "*DB2Training-impl.js*".

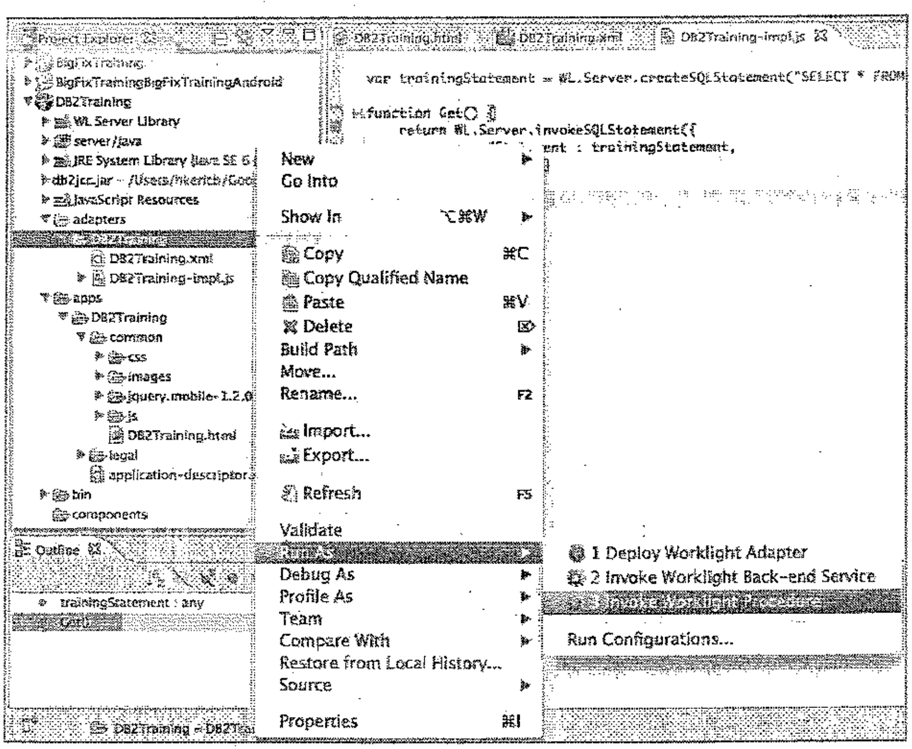


8. Remove all code and insert the following:

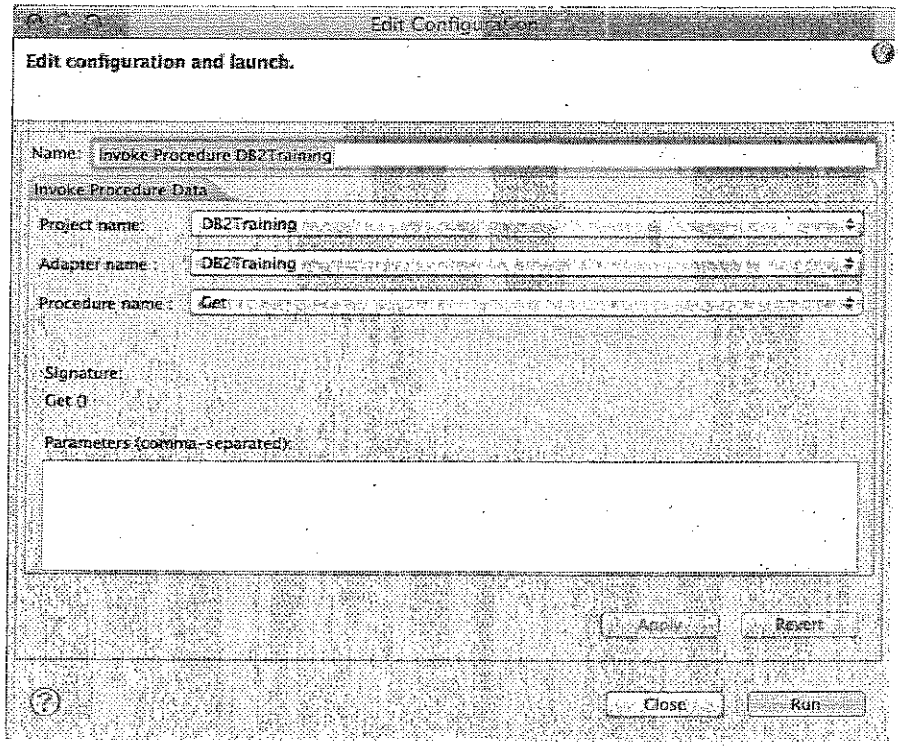
```
var trainingStatement = WL.Server.createSQLStatement("SELECT
* FROM TRAINING");

function Get() {
    return WL.Server.invokeSQLStatement({
        preparedStatement : trainingStatement,
        parameters : []
    });
}
```

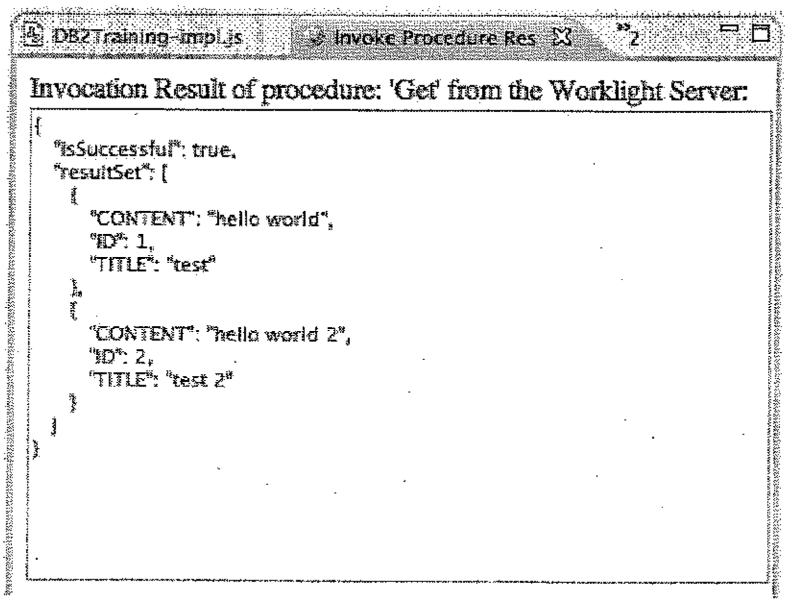
9. Now, try if the adapter can retrieve response from DB2 Server.



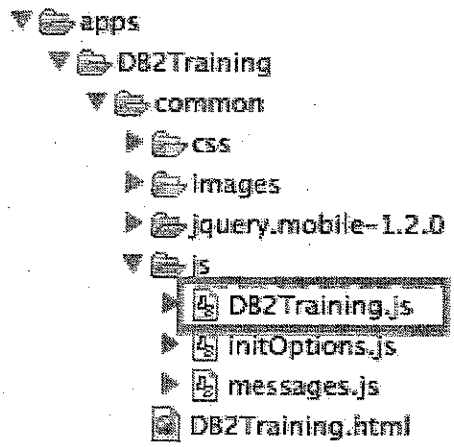
10. Choose the corresponding project name (i.e. "*DB2Training*") and adapter name (i.e. "*DB2Training*"). Choose "*Get*" as the procedure name to invoke. Keep parameters empty and click "*Run*".



11. The result should be somehow in the following format.



12. Then you can go on to integrate the results into the mobile application. The application layout interface has already been provided. You should write code to display the results. Open *"DB2Training.js"*.



13. In the *get()* function, create an *invocationData* object for the procedure call with the following format:

```
var invocationData = {
        adapter: 'DB2Training',
        procedure: 'Get',
        parameters: []
};
```

Fill in corresponding information for the procedure call.

14. Create an *options* object to define success and failure handlers.

```
var options = {
        onSuccess: getSuccess,
        onFailure: getFailure
};
```

15. Create a Worklight Procedure call.

```
WL.Client.invokeProcedure(invocationData, options);
```

16. Now you can test the application.

IBM
Programming Contest 2013

Chapter 1: Contest Environment walkthrough

Objectives

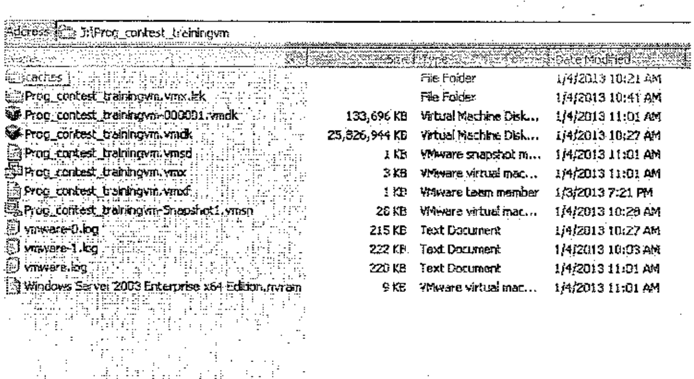
In this exercise, we will learn:

- How to boot up the programming environment using VMware
- How to log in and start up necessary services and system software
- Examine the sample application which will be used in the programming contest
- How to perform programming using the GUI environment
- How to use eclipse and create, import and export project

Exercises

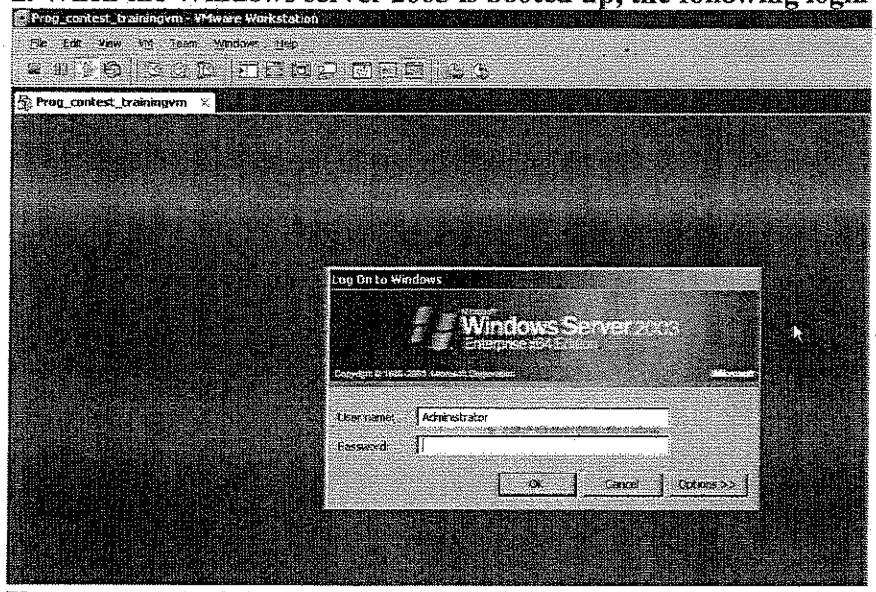
Part1. Open vmware

1. Using the Explorer, navigate to the VM folder and the double click the VMware virtual machine configuration file with name "*Prog_contest_training.vmx*" and then boot up the VMware.



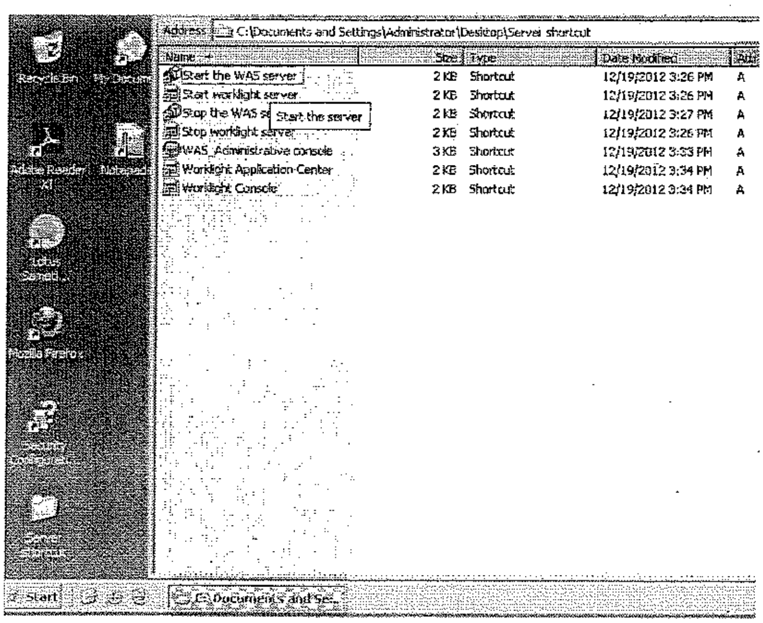
IBM
Programming Contest 2013

2. When the Windows server 2003 is booted up, the following login screen appears:

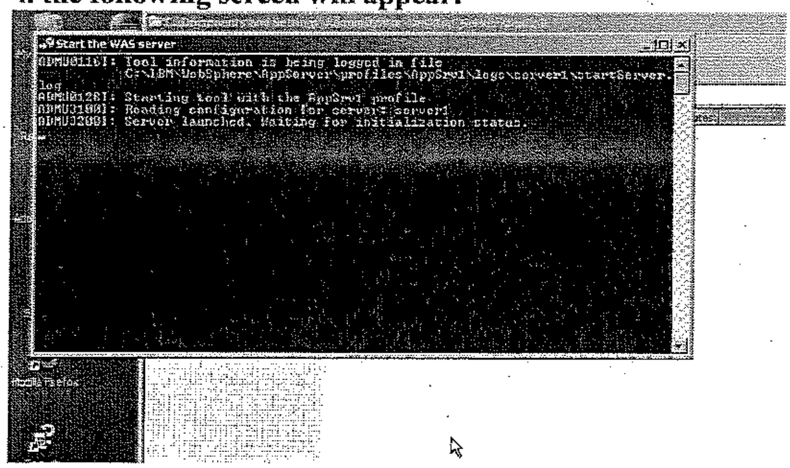


User name: administrator
Password: password

3. after login, firstly we open the folder "*Server shortcut*" on the desktop. Then we click the file "*Start the WAS server*" to start the Websphere Application Server.

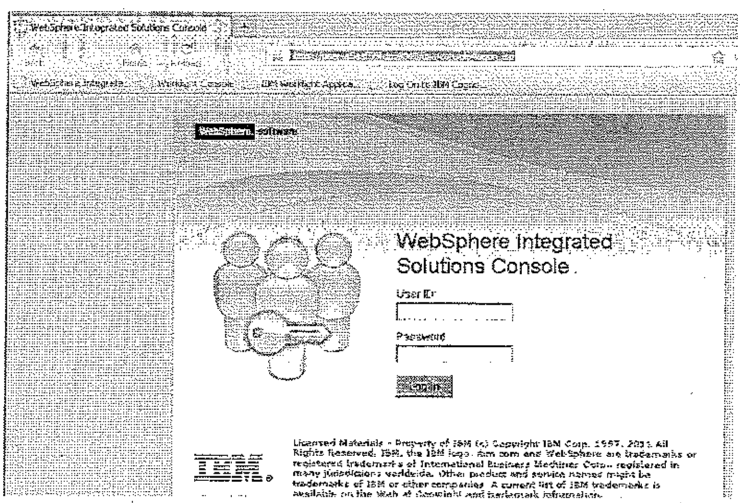


4. the following screen will appear:



We can see that Server launched. The screen will automatically close when the web server start successfully.

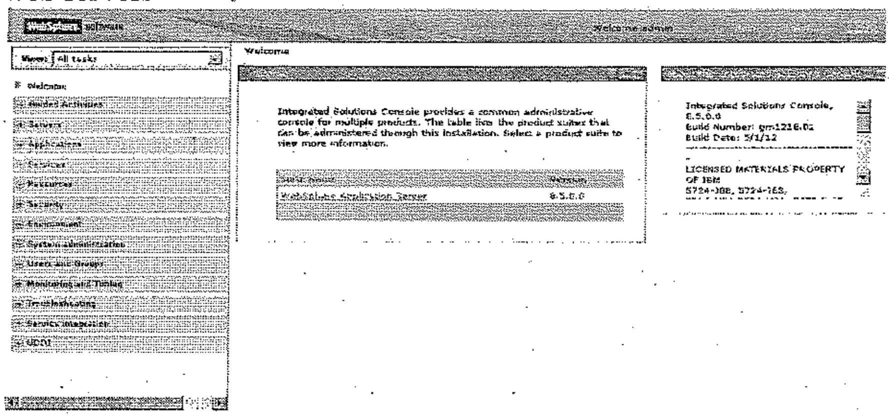
5. To test whether the web server started successfully, go the following *URL*: *https://localhost:9043/ibm/console/logon.jsp*



User name: admin
password: password

IBM
Programming Contest 2013

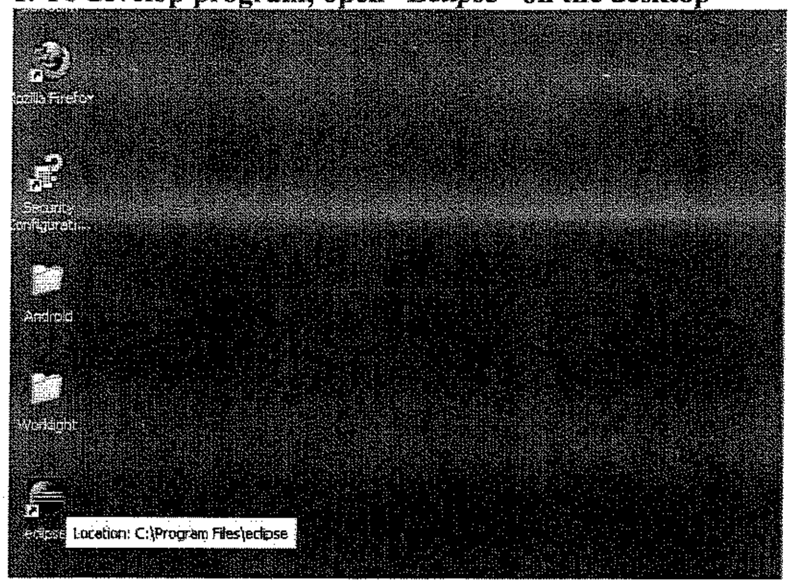
6. after login, the following screen will be shown. The page is used for setting the web server.



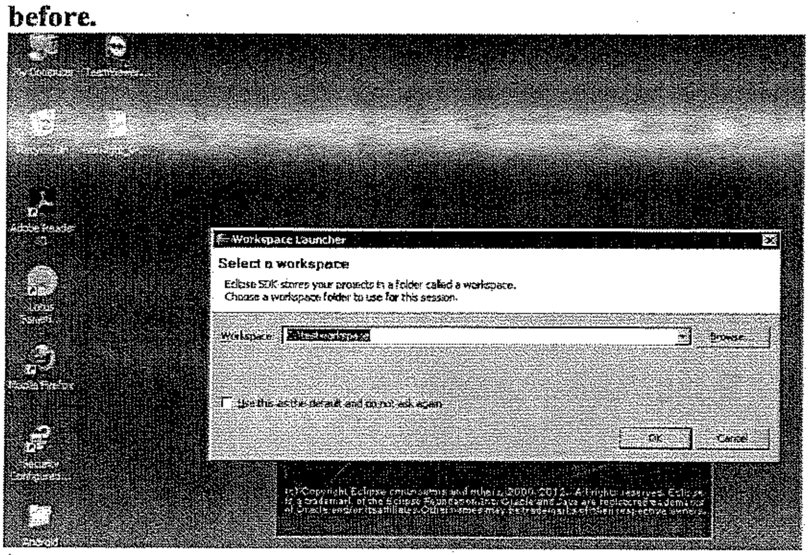
IBM
Programming Contest 2013

Part2: Open Eclipse and new project

1. To develop program, open "*Eclipse*" on the desktop

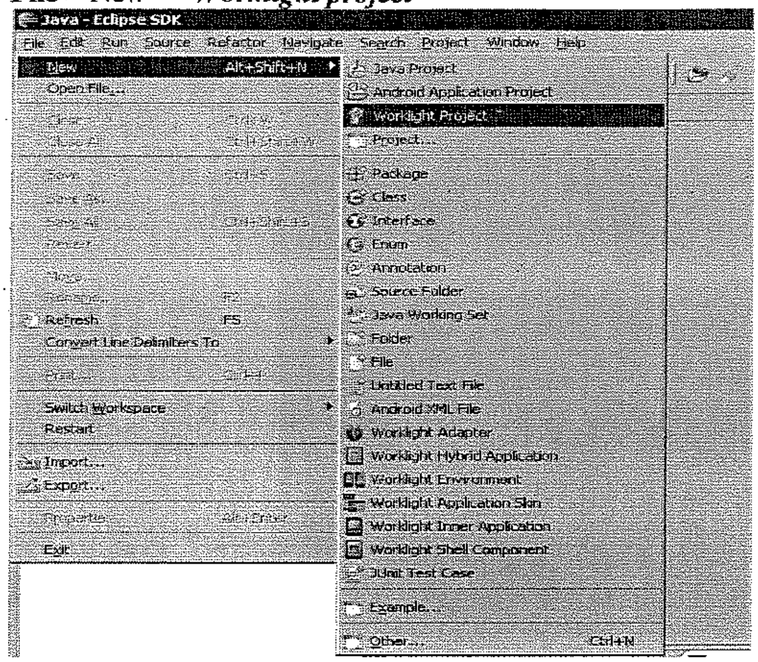


2. Select the workspace path. The workspace stores the projects that you developed before.

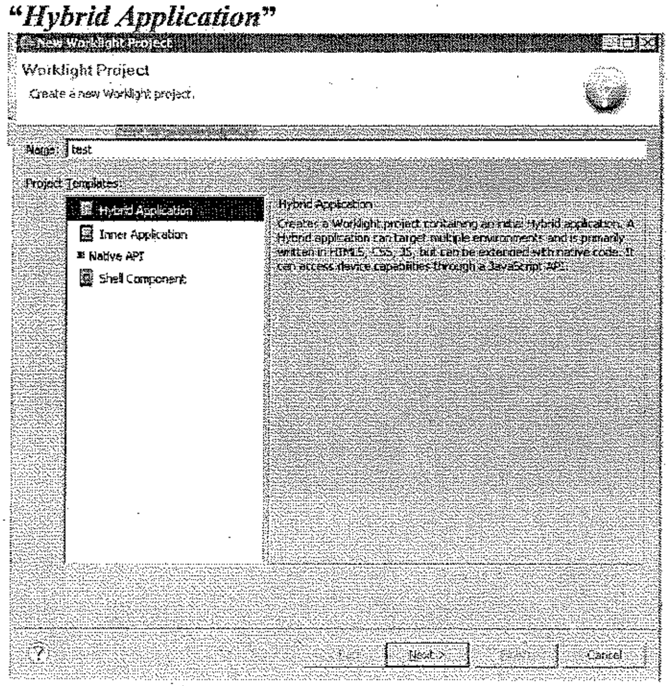


IBM
Programming Contest 2013

3. Create the new worklight project on eclipse.
File > New > "*Worklight project* "

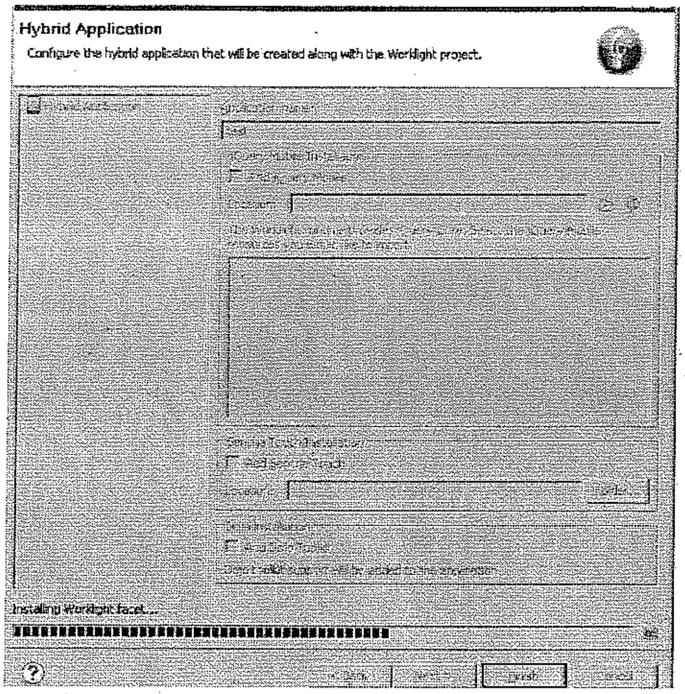


4. When creating the worklight project, choose the project templates. Choose "*Hybrid Application*"



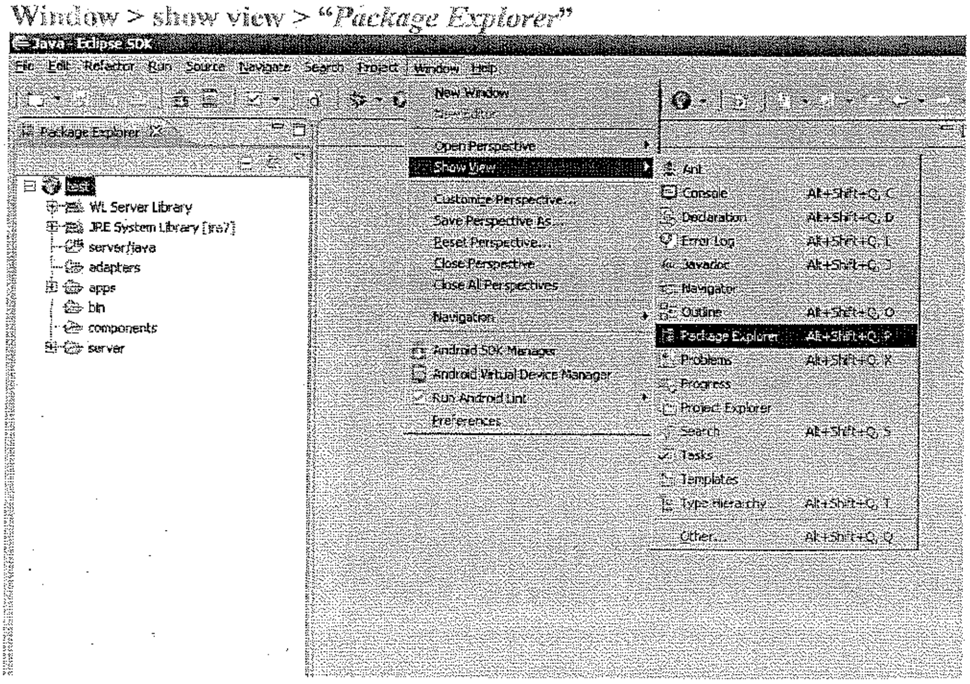
IBM
Programming Contest 2013

5. Type application name and press finish



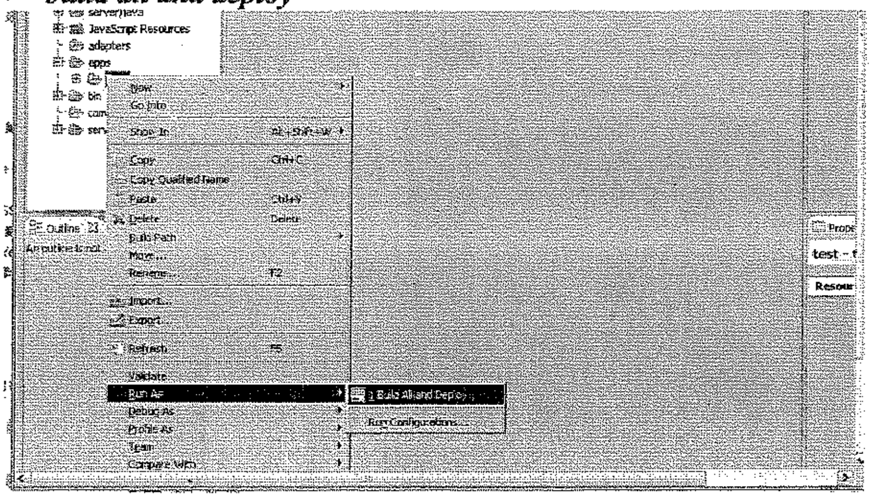
6. After creating worklight project, the package explorer shows your project on the left. If there is no package Explorer, you can go

Window > show view > "*Package Explorer*"

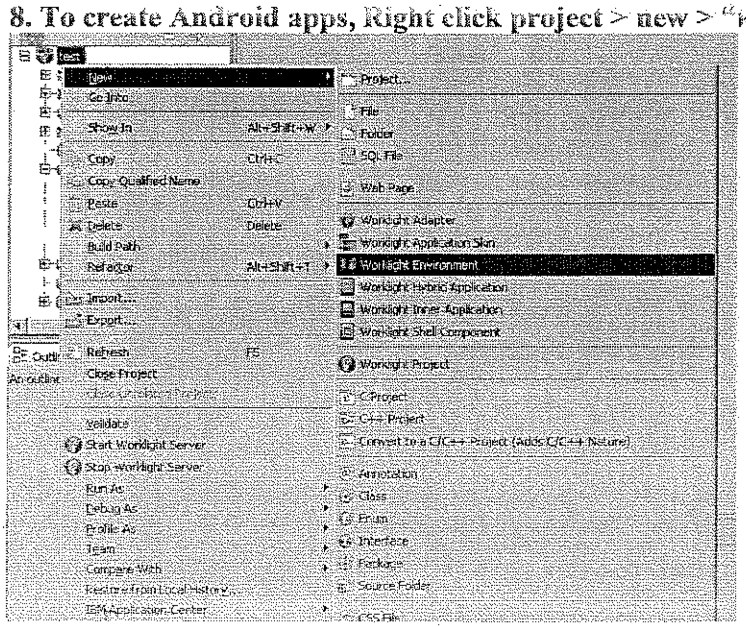


IBM
Programming Contest 2013

7. To test the project which runs properly, Explore apps > test > right click > run as > "*build all and deploy*"

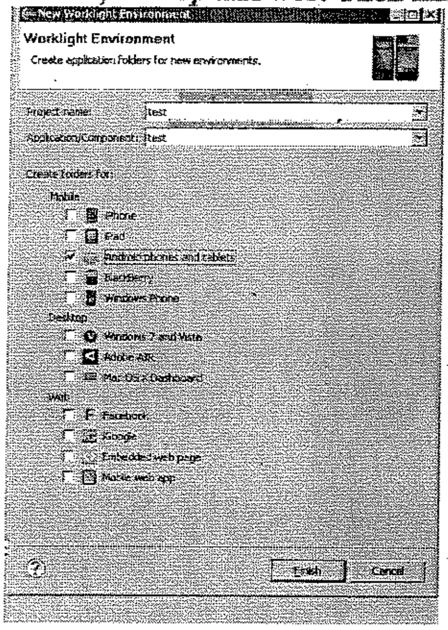


8. To create Android apps, Right click project > new > "*worklight environment*"

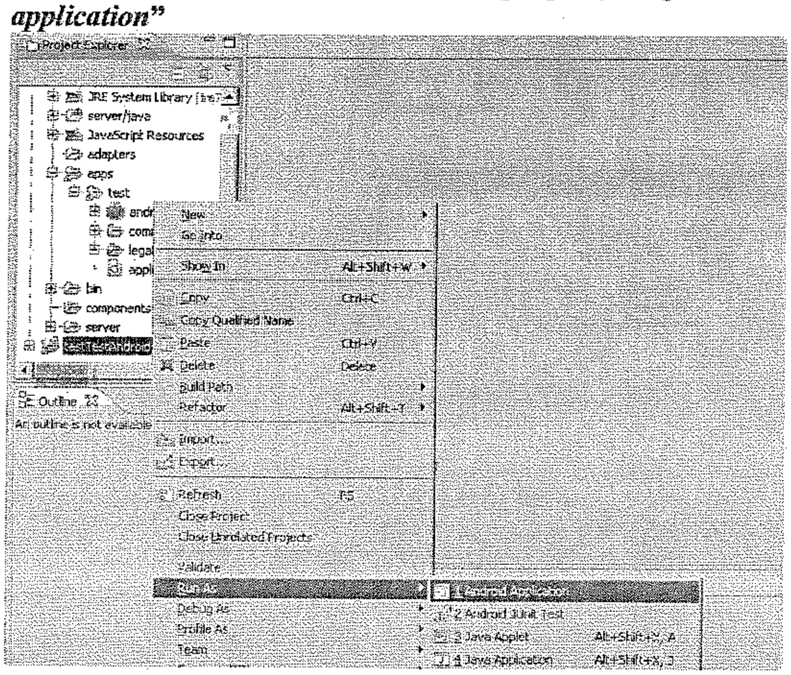


IBM
Programming Contest 2013

9. Tick the new environment you want to add. The new environment includes *Mobile*, *desktop* and *web*. Then finish.

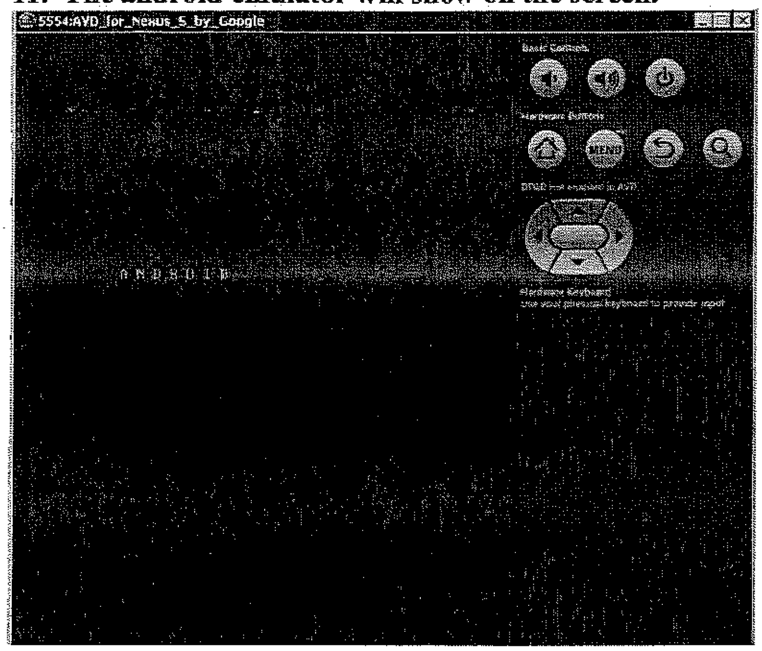


10. To test the new environment runs properly, Right click > run as > "*android application*"

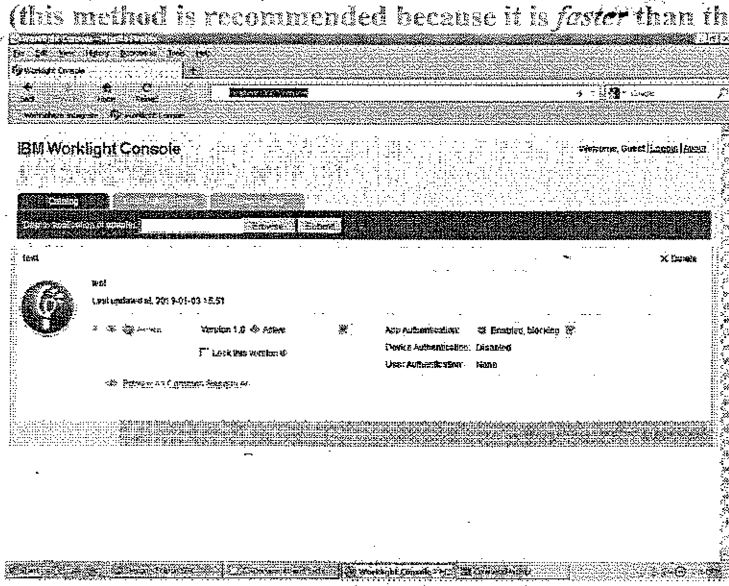


IBM
Programming Contest 2013

11. The android emulator will show on the screen.



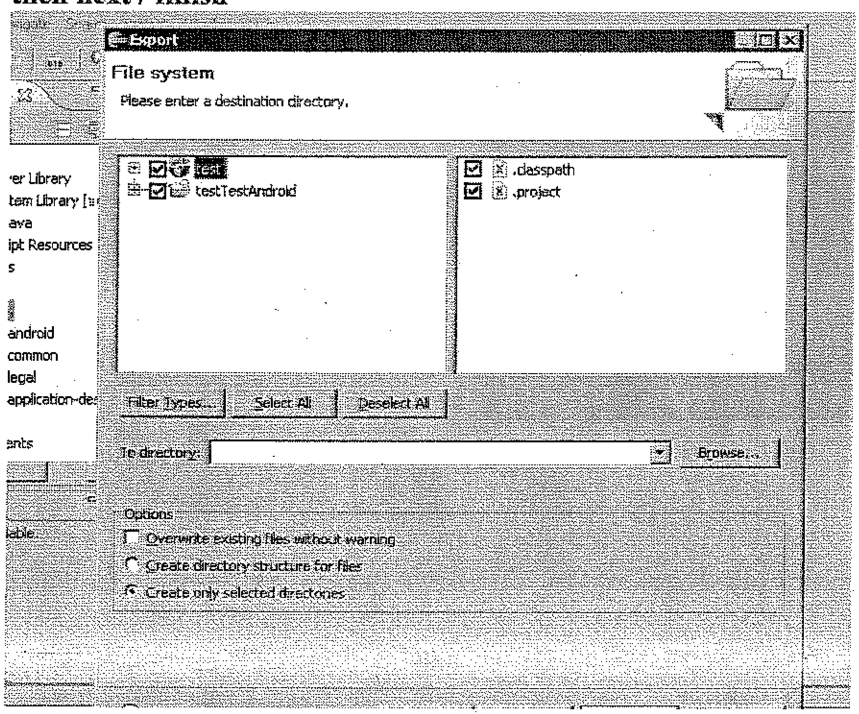
12. Another way to test the new worklight environment is to open the simulator in the web console by the following *URL:*
http://localhost:8080/console **or** ***http://IPaddress:8080/console***
(this method is recommended because it is *faster* than the emulator)



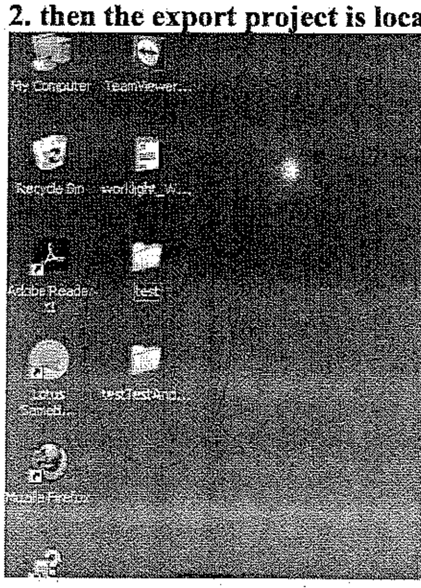
PS. If the console does not include anything, please try the step 7 again

IBM
Programming Contest 2013
Part3: Export project

1. When you want to export the whole project out of eclipse, the eclipse provides the export function.
The step is:
In the eclipse, choose File > Export > General > "*File System*", and choose directory, then next / finish



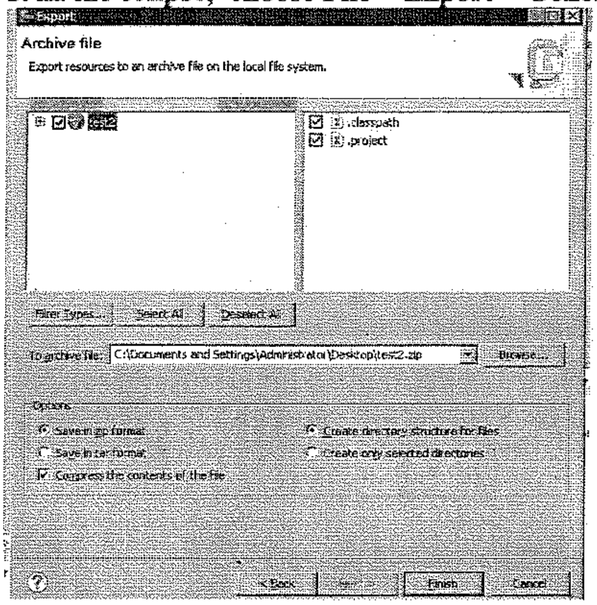
2. then the export project is located on the target directory



IBM
Programming Contest 2013

**** the alternative is to export as a zip files…**

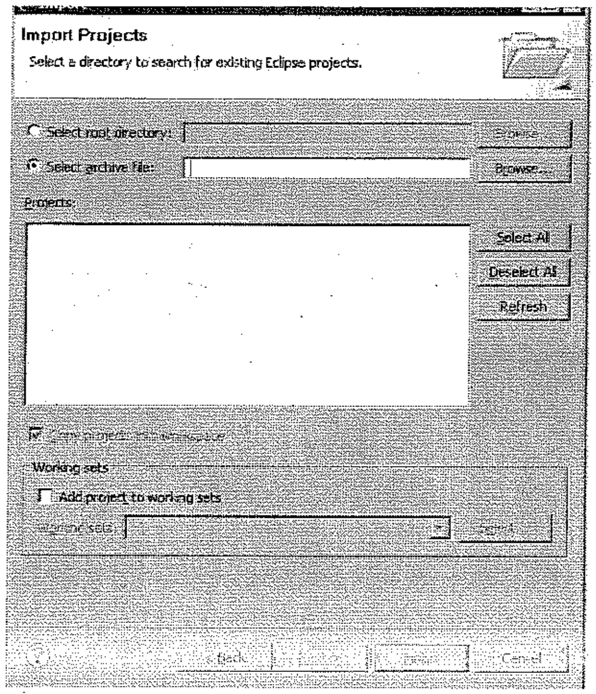
3. In the eclipse, choose File > Export > General > "*Archive File*"



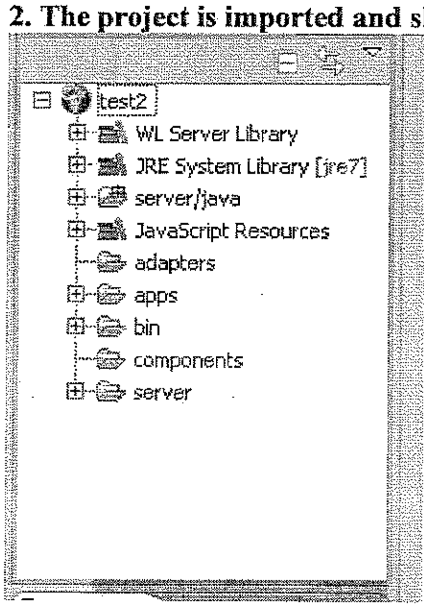
IBM
Programming Contest 2013

Part4: Import project

1. to import the project from the outside, File > Import > General > "*Existing projects into workspace*", and tick the box "*copy projects into workspace*" (Depends on which type of files you export), then next / finish

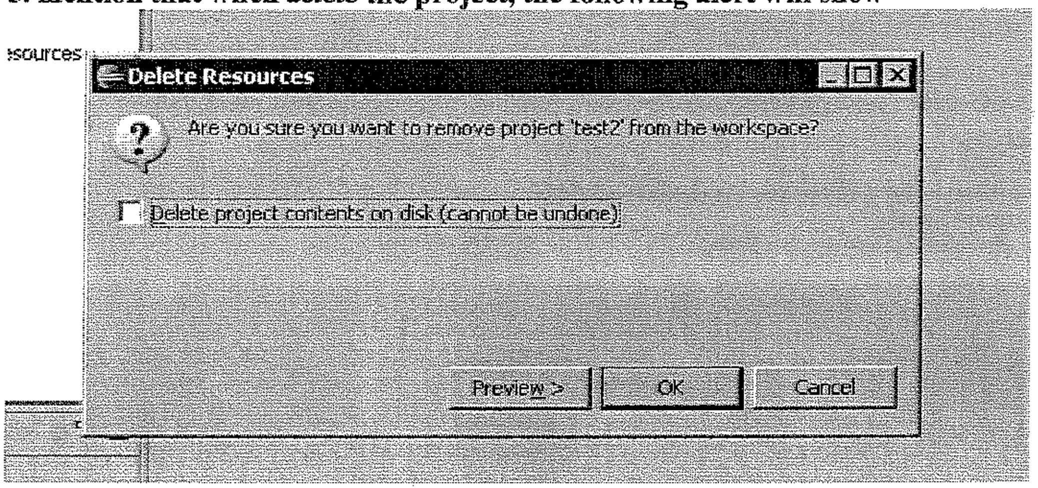


2. The project is imported and show on package explorer



IBM
Programming Contest 2013

3. mention that when *delete* the project, the following alert will show



*****If you want to *entirely delete* the project, please *tick* the box. If not, don't tick it. Otherwise all contents in the project will be deleted.**

Appendix II Multiple Choice Screen Walkthrough

Objectives

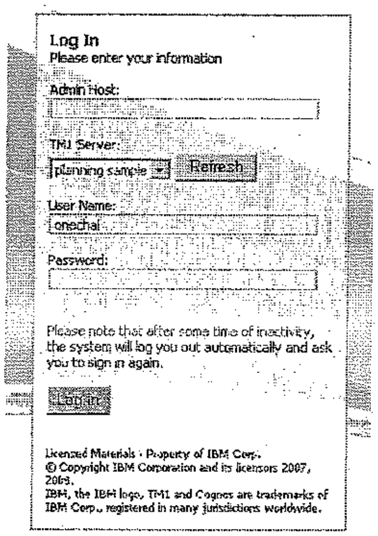
In this exercise, we will learn:

- How to connect to the multiple choice screen
- How to log in
- How to answer each multiple choice question
- How to submit your answer once you are finished
- How to examine your submission status

Exercises

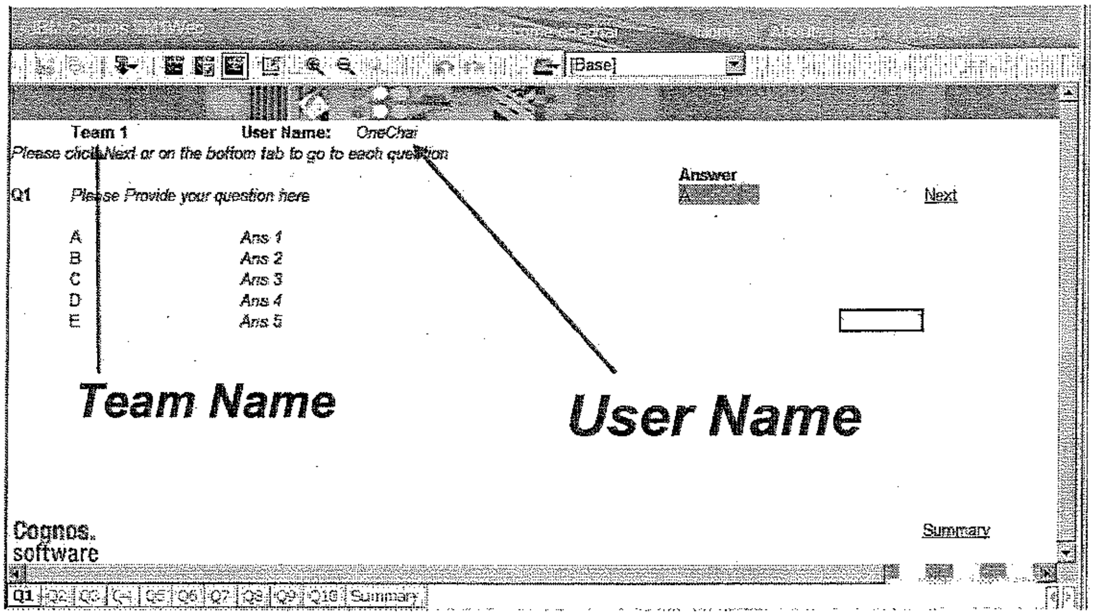
1. In the VM that is started in Chapter 1, open an Internet Explorer and type the URL http://(ip to be provided)/tm1web

   You will see the login page like this,

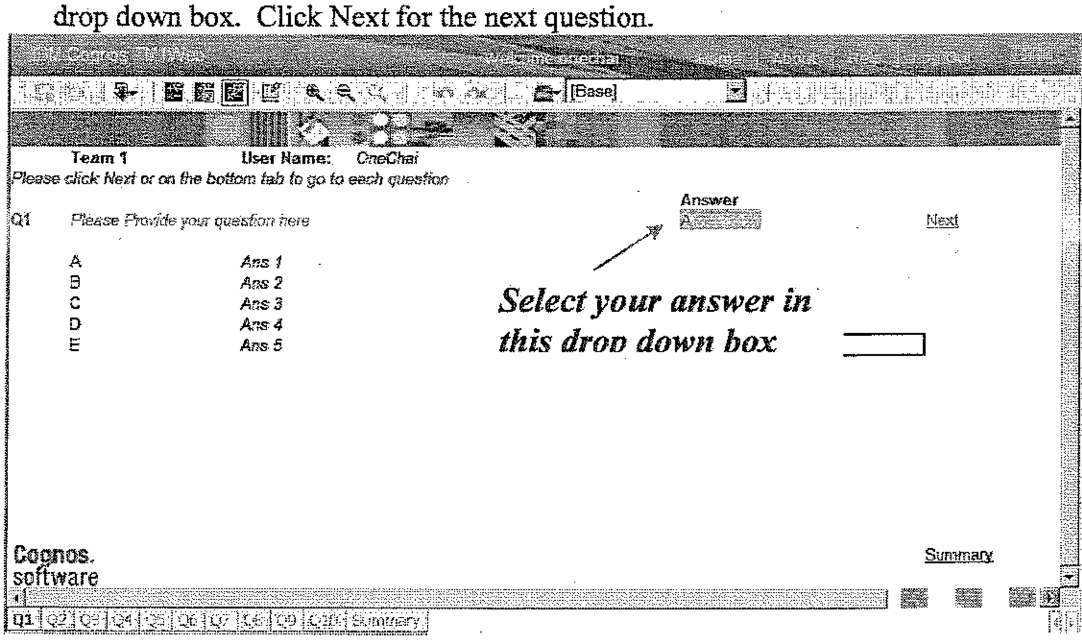


2. In the Log In screen, just leave the Admin Host blank, and select "planning sample" in the TM1 Server.

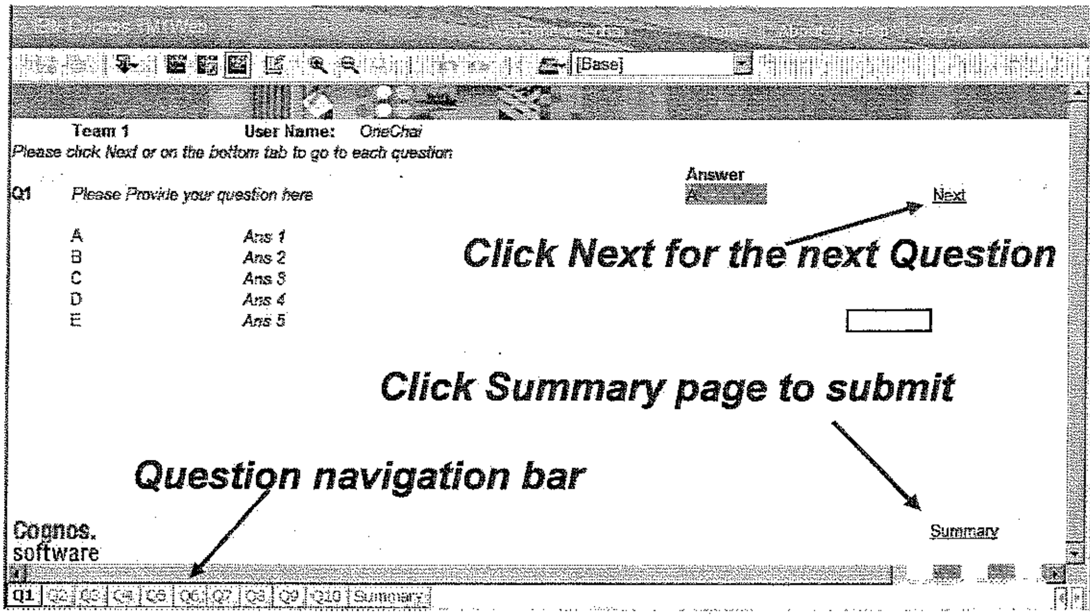
   Next, Login into the multiple choice application by the supplied credentials. After you had logged in, you can check if the team name and user name is correct.



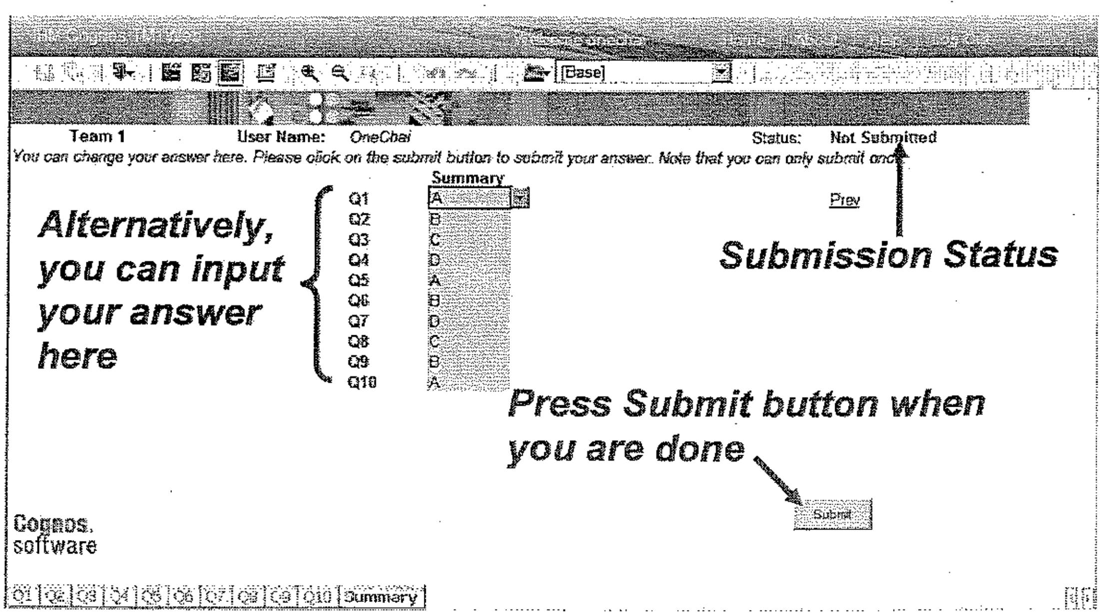
3. After logged in, the first question is displayed and selects the correct answer in the drop down box. Click Next for the next question.



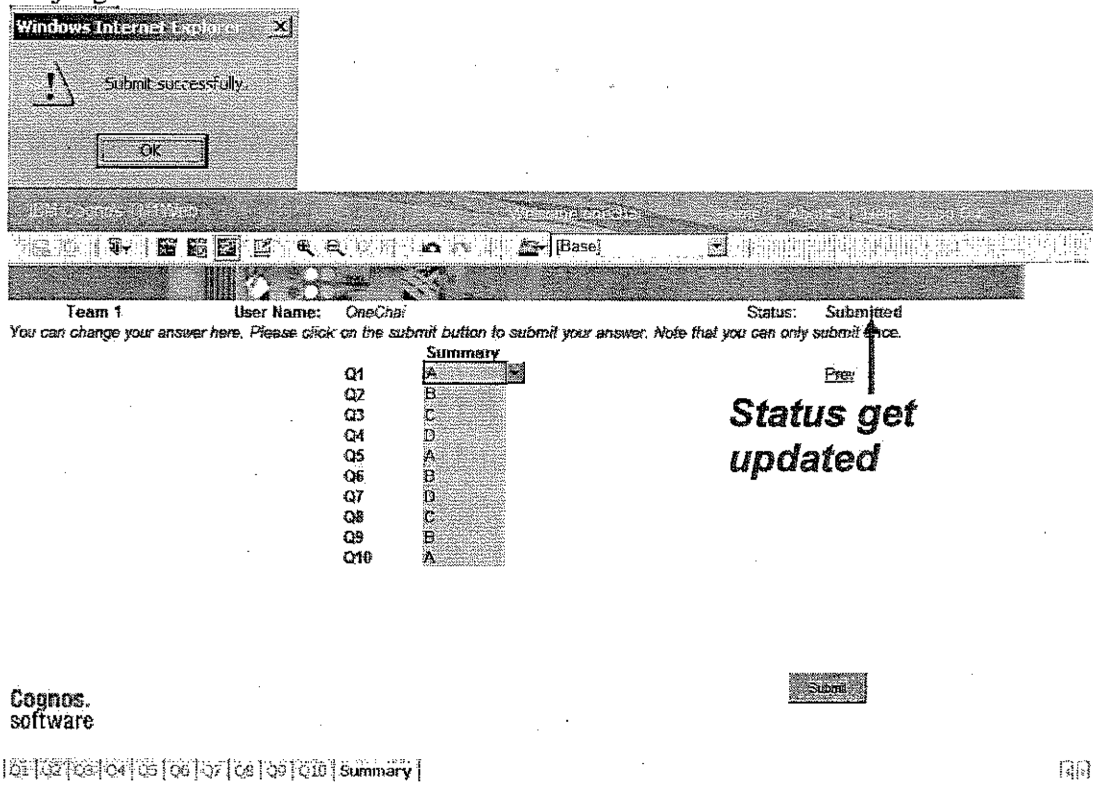
4. To navigate to different questions directly, you may access it through the bottom tab.



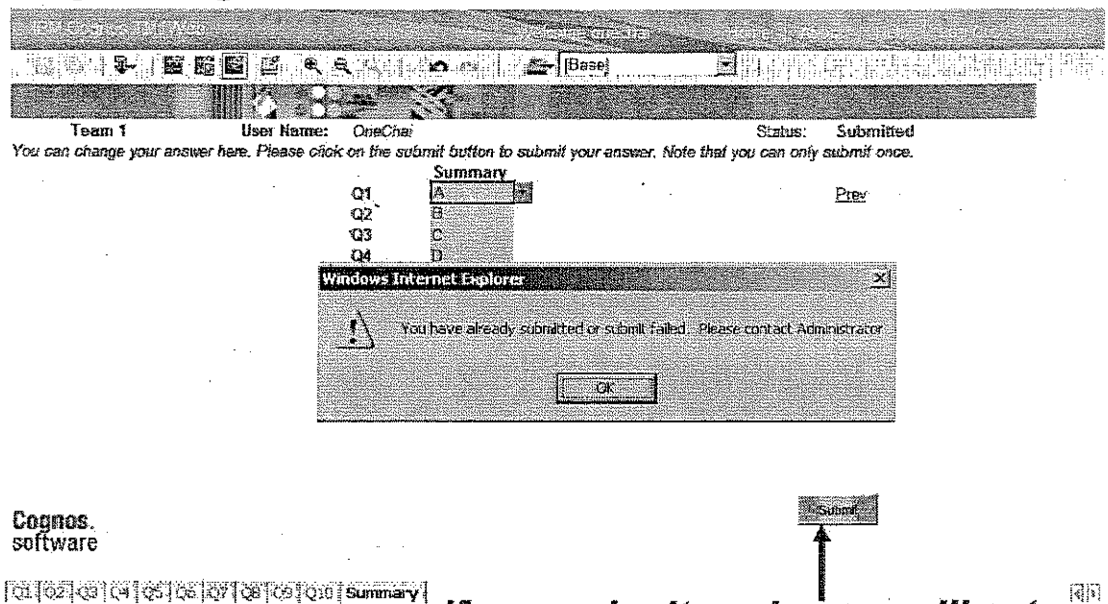
5. In order to submit your answer, you must go to the summary page and review your answer. For your convenience, you may also answer the question in this summary page directly from each of the drop down box besides the question number. Note that your Status now should be "Not Submitted". After you confirm it, press the button "Submit".



6. Once you submitted, a confirmation box should be appeared. Also check the status should be changed to "Submitted". If you got an error message, please contact your judge.



7. If you try to resubmit your answer, you will get an error message like this.



If you submit again, you will get error message