Building a Database-Enabled Web Application from Scratch -- 1

# Purpose of Exercise

This exercise integrates everything we have studied in the course. You can use the concepts and steps described here to complete the bulk of the course project. The document walks you through the process of building a database-enabled web application from scratch. The broad outline is:

1. Use Oracle SQL Developer Data Modeler to build the ERD.
2. Use Oracle SQL Developer Data Modeler to generate the database schema (SQL file containing code to build the tables and constraints) or database script based on our ERD.
3. Upload the database script to a web-based tool called Oracle Application Express (APEX) and run the script to generate the (empty) database tables on the APEX server.
4. Use APEX to build a fully functional web application around our database.

You should work through this document very carefully in order to understand the details and master some important skills and concepts. Do not just complete the steps without understanding what is going on. That will miss the point. You will need to understand these steps, and carry them out for your project.

# Oracle Application Express

Oracle Application Express (APEX) can be used to build small departmental applications or as a **rapid prototyping** tool. This means that the tool can enable us to quickly build a *prototype* of a database-enabled web application. We say *prototype* because the application developed using APEX tend to be slow on very large applications and therefore APEX should only be used to prototype such applications. However, it is a great tool to build and deploy small and medium sized departmental applications.

What do we mean by “small to medium?” Applications with a maximum of a few dozen tables each with around 10,000 rows of data and no more than around 20 concurrent users will perform reasonably well.

However, when companies build and deploy large applications, the process usually takes too long. If users have to wait for too long between the time they ask for a new application and the time they actually start seeing the first versions, then they could lose interest. Also, even though users have given their requirements, several communication gaps persist and will surface only when users interact with a working application. To enable users to see quick mock-ups of the application, software development teams often build *prototypes* that look and feel like the real application, but might not be as robust or as efficient. If a prototype can be developed quickly – say within weeks of discussing with users – then users can use the prototype and correct misunderstandings early on in the process rather than much later after a lot of time and money have been wasted.

We can install APEX on our laptops and build our applications, but this procedure can be cumbersome. Oracle provides a web based option wherein Oracle hosts the server and provides us a free account for web based development and deployment of our application. To keep things simple, I have selected this web based – cloud – option. Your application objects will all reside on some Oracle server and you will use your web browser (e.g., IE, Chrome, Firefox or Safari) to build and deploy your application.

This web-based **workspace** from Oracle allows us to have just a single database (with many tables). Oracle calls this as *schema*. So this web based account allows us to create just a single application – unless we can mix tables from several applications into this single database.

# Overview

Before we start getting hands-on, let us get a quick overview of all that we will be doing. The points below are not steps that you need to perform – those will come under "Detailed steps" later.

* **Browse the data model for our application** – Will use Oracle SQL Developer Data Modeler to browse the ER diagram. Within this, we will look at the entities and relationships. We will also look at how to create *Auto-increment* fields which are very useful for integer primary keys for which we want the database to automatically insert the next value (for example order numbers, employee numbers and so on).
* **Explore the main tools in APEX –** APEX helps us to develop database enabled web applications. We take a brief look at the tools in APEX.
* **Browse database objects in APEX –** APEX has its own database and also helps us to create database enabled web applications. We will take a quick look at a few default database tables that come pre-installed with APEX. We will look at the structure and at the stored data.
* **Build the application –** Once we have done all of the above, we will then build our web application.
* **Deploy the application –** After building the application you will run the application and make it available to anyone in the world with an Internet connection.

# Detailed Steps

1. **Browse the data model (ERD) for the application** **–** From Blackboard, under "Project/APEX exercise to help with the project," download the file *classicmodels.zip*. After downloading, if you are using Windows, right-click on the file and select *Extract all…* and accept the destination folder. If you are using a Mac, just double-click on the zip file. You will now see a folder called *classicmodels* containing a file called *classicmodels.dmd* and a folder called *classicmodels*. Open the file *classicmodels.dmd* in Oracle SQL Developer Data Modeler. Go to the tab containing the logical model. You should see the ERD shown in Fig 1. If you are on a Mac, then double-click the zip file and it will extract it automatically in a folder called *classicmodels*. Run the data modeler program and open the file *classicmodels.dmd* from the newly created folder *classicmodels*. Go to the logical model tab to see the ERD in Fig 1.

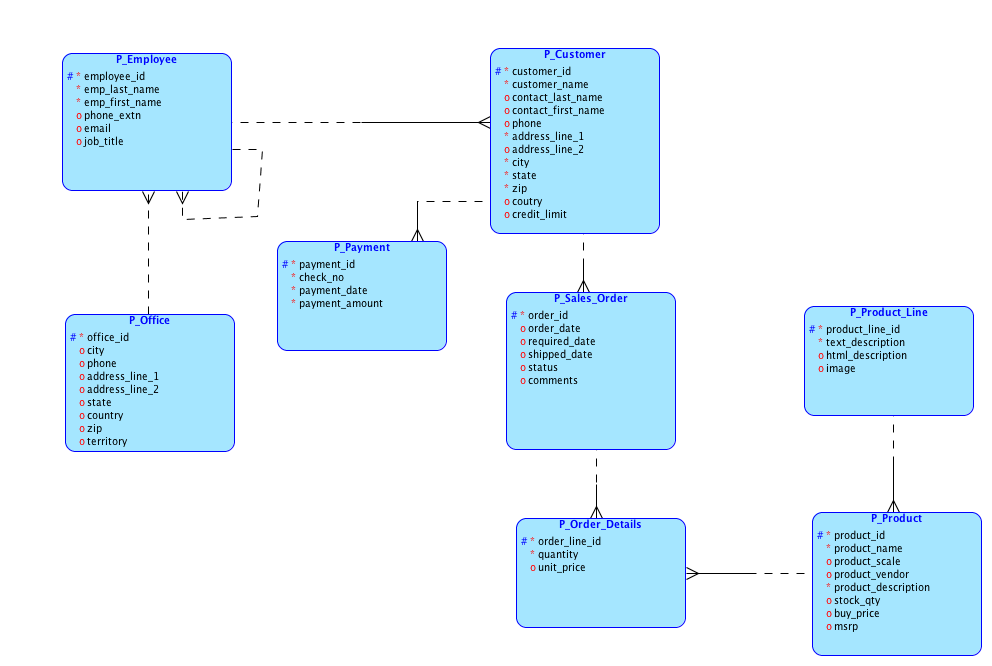


Figure : ERD for our application

Study the ERD carefully. We have prefixed every entity type with "P\_" so that the entity name does not clash with any tables that already exist in APEX. The figure shows a fairly typical business situation of a company selling many products. We have customers who place sales orders with our company and make payments. The company deals with many products, and each product belongs to a product line. For good measure we have also shown the *Employee* entity type that sports the usual unary relationship to depict the organizational hierarchy.

We can think of P\_*Order\_details* as the associative entity type between *Order* and *Product*. To simplify matters we have given P\_*Order\_Details* its own primary key *order\_line\_id*.

Note that we are not using key migration at all -- every entity type has its own primary key. **For your project please follow this approach. It makes things easier.**

Note that we have assigned a type to each attribute of every entity type. This is essential. To assign a type to an attribute, elect the attribute in the entity type properties and then select the "Logical" radio button under "Data type". After that we use the "Type" drop down box to select the appropriate type. We will use only integer, numeric (for numbers with a decimal component – like price), varchar (character fields with a given max length) and date types.

1. **Generate the relational model** – We will not go into details of what the relational model is, but you need to generate it before you can generate the SQL code for the database. To convert the logical design to a *Relational Model*,first check that a blank relational model already exists in your design. It should exist by default. See the highlighted portion of **Error! Reference source not found.**. You should see the node “Relational Models [1] as highlighted on the left. If instead you see “Relational Models []” then you might have deleted the node. In the very unlikely even that you encounter this simply right-click on “Relational Models []” and select “New Relational Model” and you will be all set.

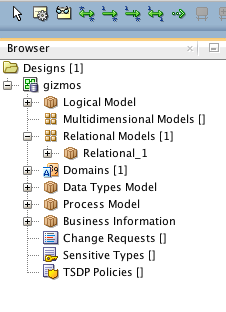


Figure : The Relational model

Now, select the tab “Logical (classicmodels)” that has your ERD and then, on top, click on the “>>” icon that will start the process of generating the Relational Model. On the resulting window click “Engineer” at the bottom. This will generate the relational diagram, which you can explore if you want. **This diagram uses a different notation than ERD and you should not confuse it for the ERD. You will see things that look like key migrations, but they are not.**

**DO NOT CONFUSE THE NOTATION THAT YOU SEE IN THE RELATIONAL MODEL WITH THOSE WE USED ON THE ERD. THE NOTATIONS MEAN DIFFERENT THINGS.**

In what follows we will do a few minor things in the relational model.

1. **Auto-increment**  **–** Each of our entity type has an integer primary key. I suggest you do the same for your project as well. Now, imagine that we are creating a new sales order. This new order would need an *order\_id*. We could manually assign it each time we create an order, but that would be cumbersome because we would have to know the order id of the previous order and then assign the next number as the id of the new order. Instead, we can ask the system to assign the one-up number automatically – *auto-incrementing*. Most databases can do this through a feature called as *sequence*. You do not need to know about *sequences* for now, but we can make a small change to our model so that the DDL code that we generate to create the database incorporates the auto-numbering feature.   
     
   You should see why this makes sense only if the primary key field is an integer field. You will not be able to do it for other types of fields.  
     
   Let us see the process to make **employee id** as an auto-increment field. Click on the "Relational 1 (classicmodels)" tab. Then double-click on the entity type P\_Employee because that entity type has the employee\_id attribute. Then click on "Columns" on the left (the second option from the top. Then double-click on the "employee\_id" attribute. The first option on the left "Genera;" should already be selected. If not select it. Then on the right hand side, check the box for "Auto increment". By default, auto increment starts from 1. If you want to change this, click on "Auto increment" on the left pane and then set your values accordingly.  
     
   You should do this now for the **primary key of every entity type**.   
     
   **Do not do this for foreign keys. Auto-increment is only for the primary keys.**
2. **Change all foreign key names** – Just as we did for the entity types, we can double click the entity types in the r**elational model** and see the columns. Select the tab "Relational 1 (classicmodels). Double-click on "P\_employee". Select "columns". In the list of columns, you will see that there is a column called "P\_Employee\_employee\_id" – this is the foreign key that was added to represent the unary 1:n relationship that P\_employee has with itself. We would of course like this column to be called "manager\_id". To do this, double-click on the column name "P\_Employee\_employee\_id" and change its name to "manager\_id". This way the column will have a better name when the tables get generated later. In similar vein, double-click on "P\_Payment" in the relational model. We see that it has a foreign key column called "P\_Customer\_customer\_id" – double-click on it and change it to just "customer\_id". Likewise, look at every table on the relational model and change the names of all foreign key coulmns (they are marked with F) and get rid of the "P\_xxx" prefix. In the Customer entity type, the foreign key should be called "sales\_rep\_employee\_id". make all these changes carefully. Save the model with File-> Save.
3. **Check that everything is OK before generating database script** – We will now be proceeding to generate the SQL file for creating the database – the database script. Before doing that, we should make sure that all is well. **Check each of the following before generating code. Save the model before proceeding to generate code:**
   1. **Your ERD correctly represents your scenario**
   2. **None of the entity type or attribute names has any spaces or hyphens in the name. Underscores are allowed.**
   3. **Every entity type has its own integer primary key**
   4. **Every primary key has been specified to be auto-increment**
   5. **There is no key migration, especially for associative entity types**
   6. **Every attribute has a data type specified – that is none of the attributes shows up as having "unknown" data type in the attribute list of its entity type.**
   7. **For every varchar attribute, you have specified a size.**
   8. **For every numeric attribute (not integer), you have specified a precision (total number of characters) and scale (number of characters after the decimal point.**
   9. **You have changed the names of every foreign key column to something meaningful.**
   10. **You have made the primary key of every entity type to be "auto-increment."**
4. **Warning – If you generate the relational model anew (do the "engineer" step again, then you will lose all the changes you made to the earlier relational model – like auto-increment and changes to all foreign key names. You will have to redo these.**
5. **Generate the SQL file for the** database -- Now that we have completed the ERD, assigned types to every attribute (including the size for every VARCHAR field and precision and scale to every numeric field (not Integer field), and we have set up auto-increments for the primary key of every entity type, we are ready to ask the system to generate the SQL code to create our database.

Select either the "Logical (classicmodels)" or the "Relational 1 (classicmodels)" tab. Then use the menu option "File -> Export -> DDL file." On the resulting window, click "Generate". **On the next window select the tab "Drop selection" and check all the options** – this will save you a lot of trouble if your= have to re-load your script to APEX. Then click "OK" – if you have done everything as suggested, you will see an error message that there is one error – don’t panic.

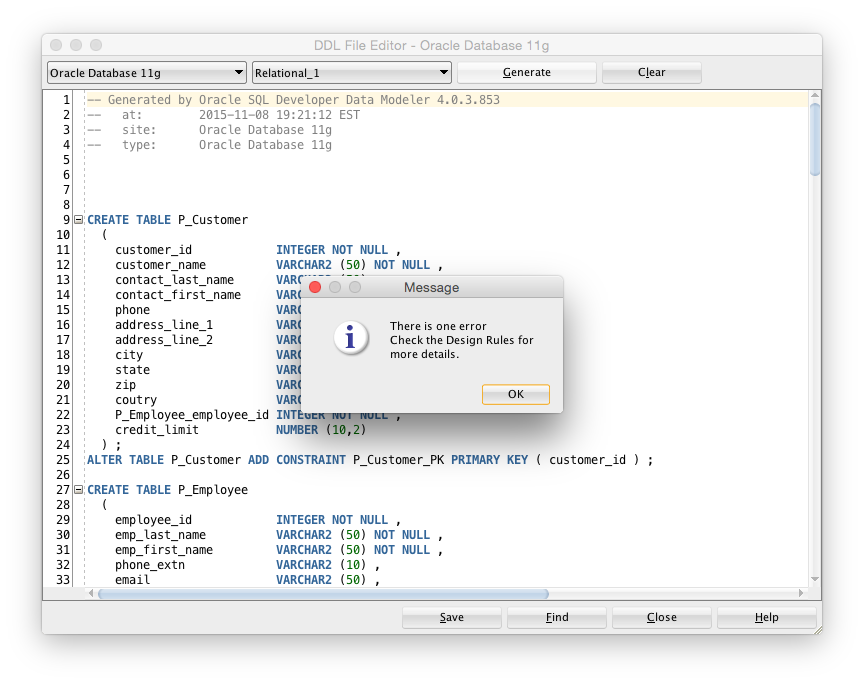


Figure : Code generation error – don’t panic!

1. From the earlier handout "ODM step by step" we already know that this error is most likely because the system generated too long a name for a foreign key. Click "OK" and scroll down towards the bottom of the file to find the line with " -- ERROR: FK name length exceeds maximum allowed length(30)". This should be around line 117. The name " P\_Order\_Details\_P\_Sales\_Order\_FK" is longer than the allowed 30 characters. Change that name to something shorter like "P\_Order\_Details\_Order\_FK" – be sure that the name does not contain a space in it. Once you have fixed all such errors, save the file. Name it as "classicmodels.sql". Save it in a known place where you can find it later. You will need this file.  
     
   Take a look into this file. The code in there should mostly be familiar to you. Other than things to do with sequences, we have looked at everything else.  
     
   We will now start looking at the APEX end to build a web based application for our domain.
2. **APEX Tools** – Using the credentials and the URL that you got from Oracle, login to your APEX workspace. Once you login, you see the screen shown in Figure 5.

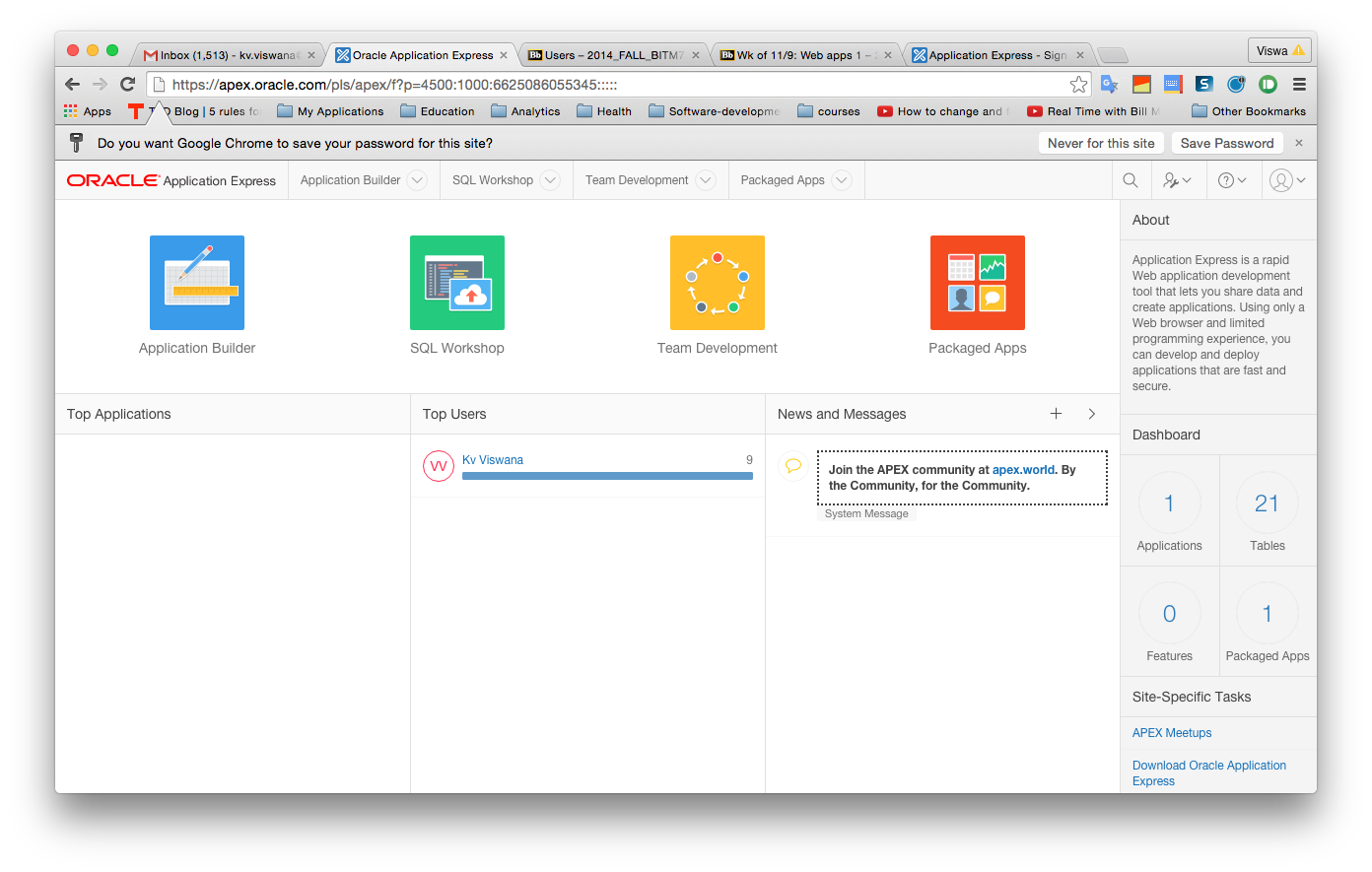


Figure : APEX initial screen

* 1. You see that APEX has several tools. We will use the following:
     1. **Application Builder** helps us to – you guessed it – build applications. More seriously, we use the application builder to create various screens and connect them together. Each screen typically interacts with the underlying database to fetch or update data in one of more tables.
     2. **SQL Workshop** enables us to manage the various objects in the database. We can manage individual table definitions; create keys and constraints and so on.

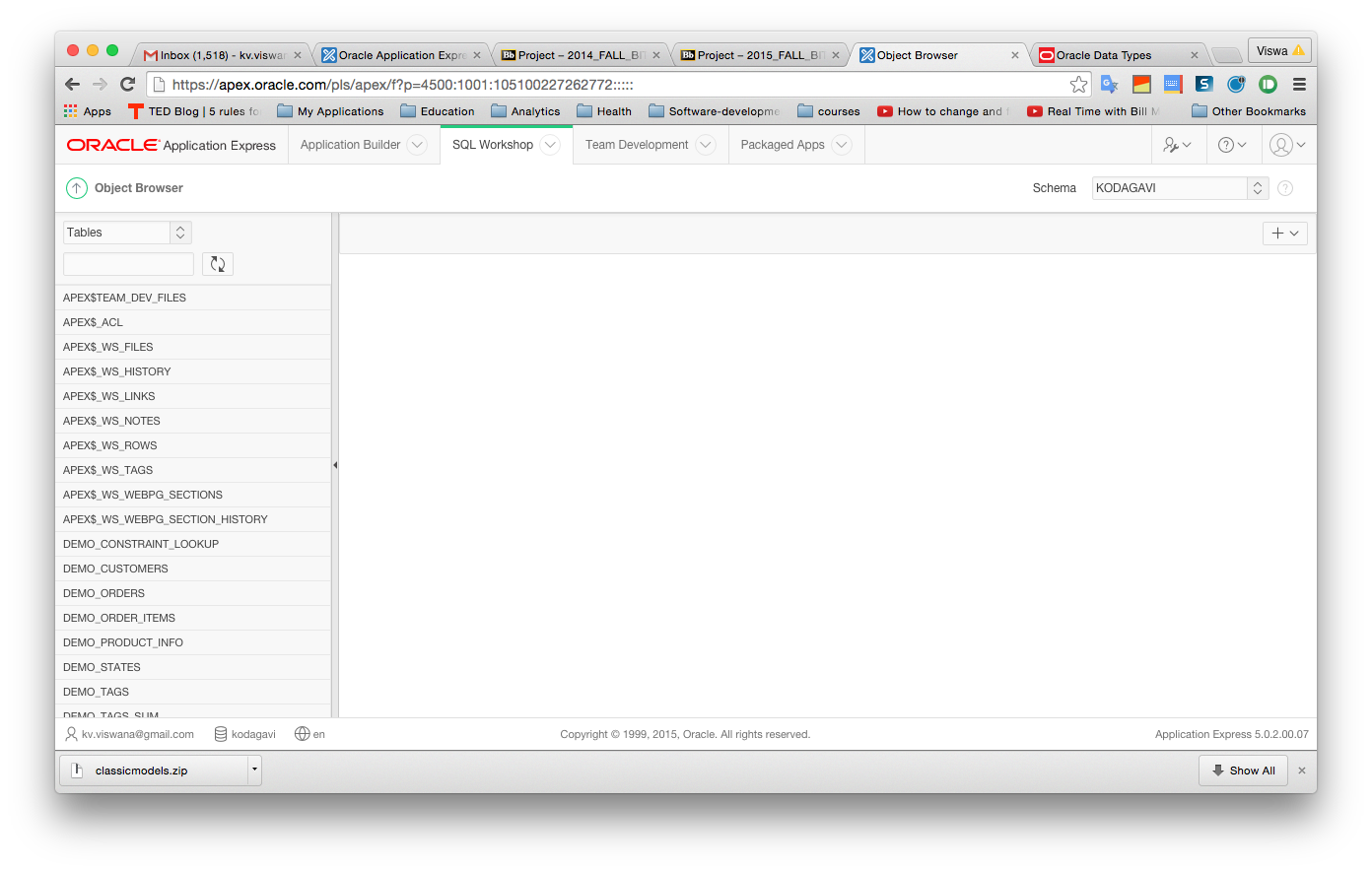
1. **Browse the database** –Dropdown the menu item *SQL workshop* and select *Object browser*. This will enable us to see the objects currently in the database. You should see something like Figure 6. By default you see the tables in the database, but you can see other objects by selecting the dropdown button in the top left of the window. We did not create these tables; they exist by default. APEX uses some of these tables for its own working and a few tables like *Emp* and *Dept* are for use in demos.  
     
   

Figure : Browsing APEX database objects

1. **Browse tables** – APEX comes with some demo tables by default. *Emp* is one such. Click on *Emp* in the list of tables on the left. This will bring up the definition of the *Emp* table. You can see that the table has many columns; you can see the definition for each column. You can click on *Data* above the table to see the data in the table. You see the data for 14 employees. Explore some other tables. It is now time for us to load our application’s data to the database on the APEX workspace.
2. **Create the database and load the data for our application onto our APEX workspace –** In steps 5 and 6 above that we generated the SQL script to create the database tables. Before we can execute this SQL script on the APEX workspace, we will first need to upload the script to the APEX workspace and only then execute them to create the tables.
   1. **Upload the script files to the server –** We will now first upload the database script file to our APEX workspace. In APEX, expand the menu item *SQL Workshop* and then select *SQL scripts*. Then click on *Upload*. Then click on the *Choose file* button and then select the file *classicmodels.sql* that we saved in step 6. earlier. In the field *Script name*, you can enter any name. Enter classic\_models\_*schema* and then click *Upload*. At this point, the script is on the APEX server, but we have not yet executed the script to create the database tables. We will do that in the next step.
   2. **Execute the script –** If you are not in the *SQL Scripts* view already, go there by dropping down the *SQL workshop* menu item and selecting *SQL scripts.* You see the script that you just uploaded. Notice the column titled *Run* with an arrow button. Click on the *Run* button corresponding to the script *classic\_models\_schema*. On the next screen click on *Run now*. Once this completes, you will be taken to a window that shows the status after executing the script. You can click on the lens icon in the "View results" column. If you scroll down, you will see on the bottom if there were any errors. If you did everything right, there should be no errors.
   3. **Browse the tables and data –** Now that we have created the tables and loaded the data, let us take a quick look at what we have accomplished. Drop down the menu under *SQL Workshop* and select *Object Browser*. You should now see the tables *Customer, Employee, Office, Sales\_order, Order\_details, Product\_line, Product* and *Payment*. Browser these tables and note that we have lots of data in some of the tables. You might find it useful to compare the tables and attributes here with what you have in the ERD. (They should match, but please let me know if you see any discrepancy.)

Now that all our tables and the data are in place on the APEX workspace, we are ready to weave the final step of the magic.

1. Before we jump into the steps for creating the application, let us take a quick look at what we plan to build. Figure 6 shows the login page.

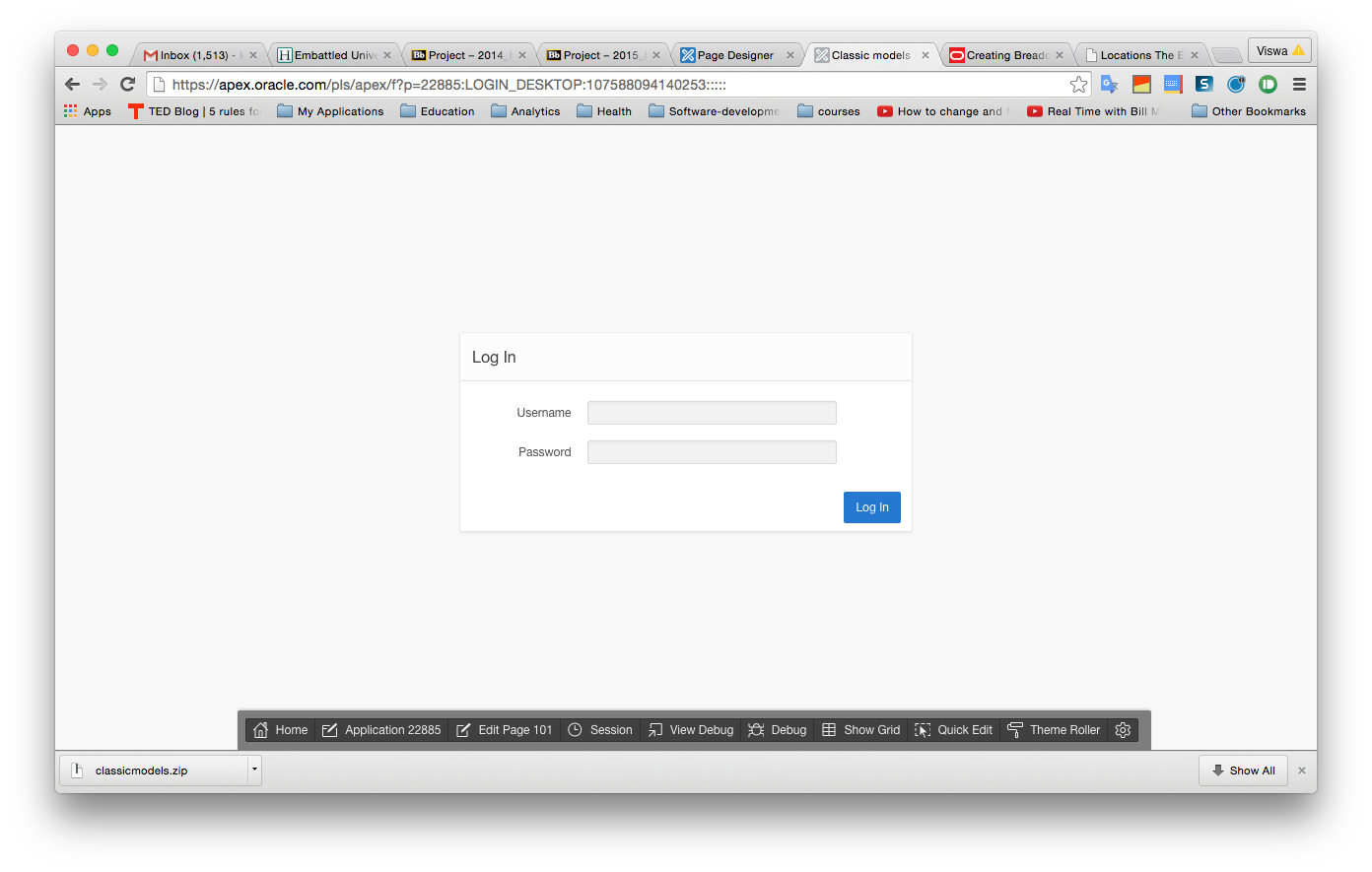


Figure : Login page

1. Once the users log in with a user account, they will see the main screen shown in Figure 7:

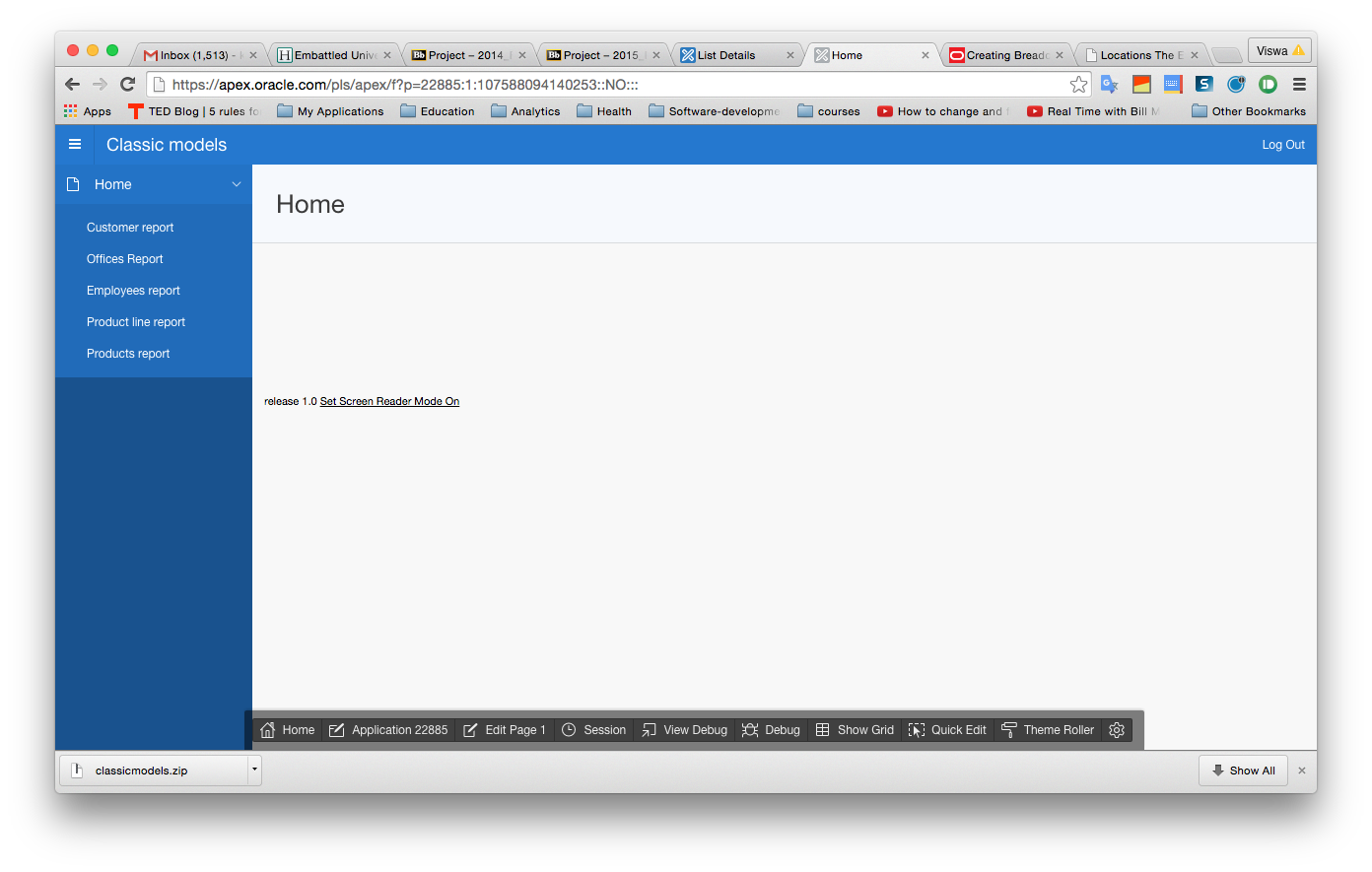


Figure : Starting screen for the application

1. The application has a set of tabs on the left. Users can select the appropriate tab for their desired action. For example if users what to maintain customer information, they will click on the *Customers* *Report* tab. We will not aim to do anything very complex in this application. APEX has lots of features and we can build very sophisticated applications. However, we will only be building a simple application to enter data into the various tables and create and view sales orders.
2. If users click on *Customers Report,* they will see the screen in Figure 8. Currently we have no customers. You can click on "Create" near the top right and create a new customer. However, as per our model, creating a new customer requires us to provide an employee\_id who is the sales rep for that customer. We cannot do that until we have created employees.
3. Once we have some customers, they will show up on this report and users can click on the edit icon in the leftmost column of each row and edit the customer on that row. This produces a new screen that allows editing. Users can also use the *Create* button on the new screen to create a new customer. The column titles on the report page are also active and you can click on them to sort and filter the report display.
4. **Creating the application –** Drop down the list under *Application Builder* and select *Database applications.* You should see the screen in Figure 10. It shows that we currently have only one database application in our workspace – Sample Database Application. If you want, you can click *Run* and run the application. You will need to login to the application with your APEX credentials. If you do run the application, click on *Home* at the bottom and then click on *Application Builder* once you are ready to proceed. This will bring you back to the listing of our database applications.

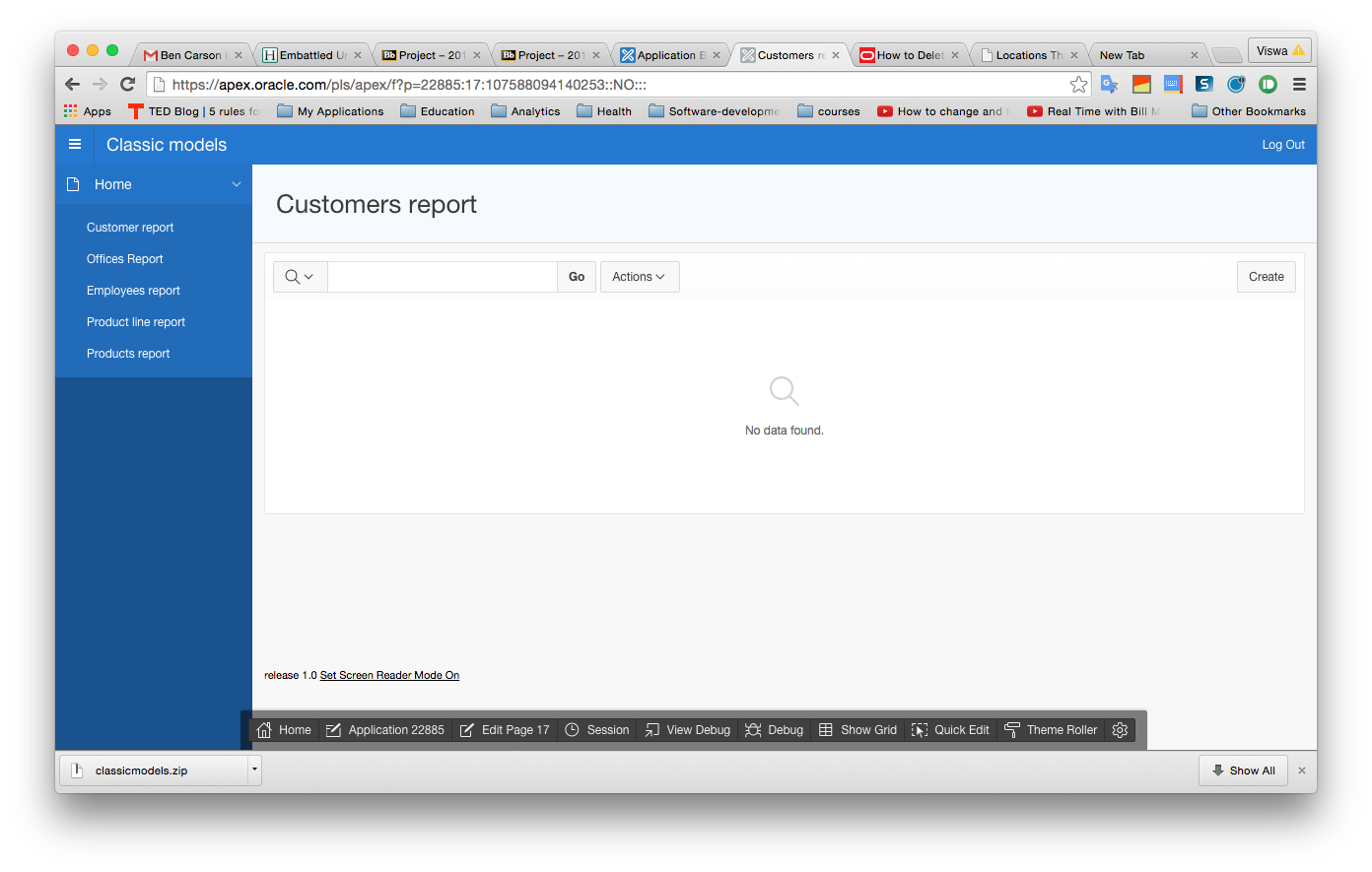
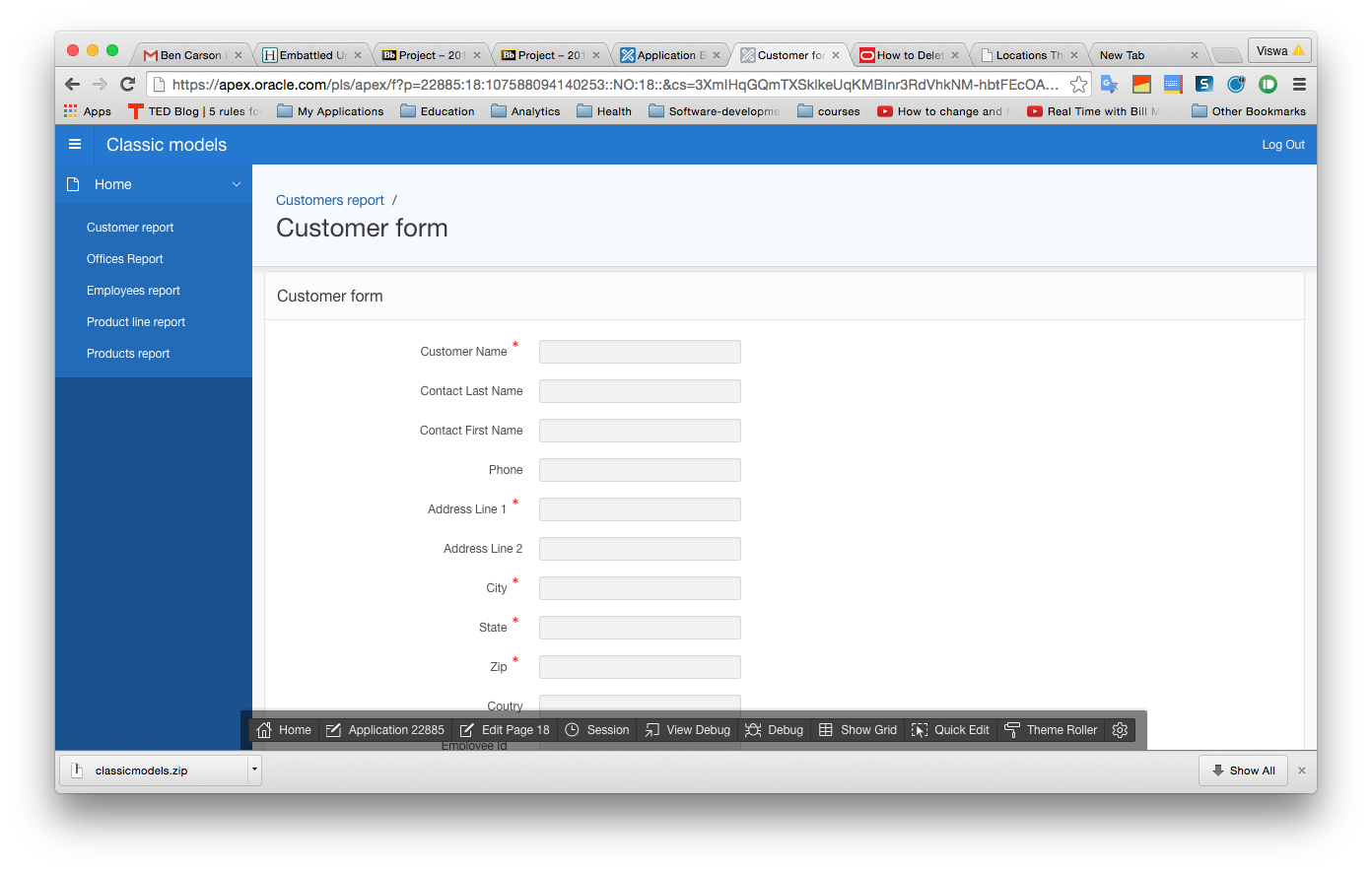


Figure : Customer report



*Breadcrumb*, showing the navigation path to current screen.

Figure : Breadcrumb showing navigation path to current screen

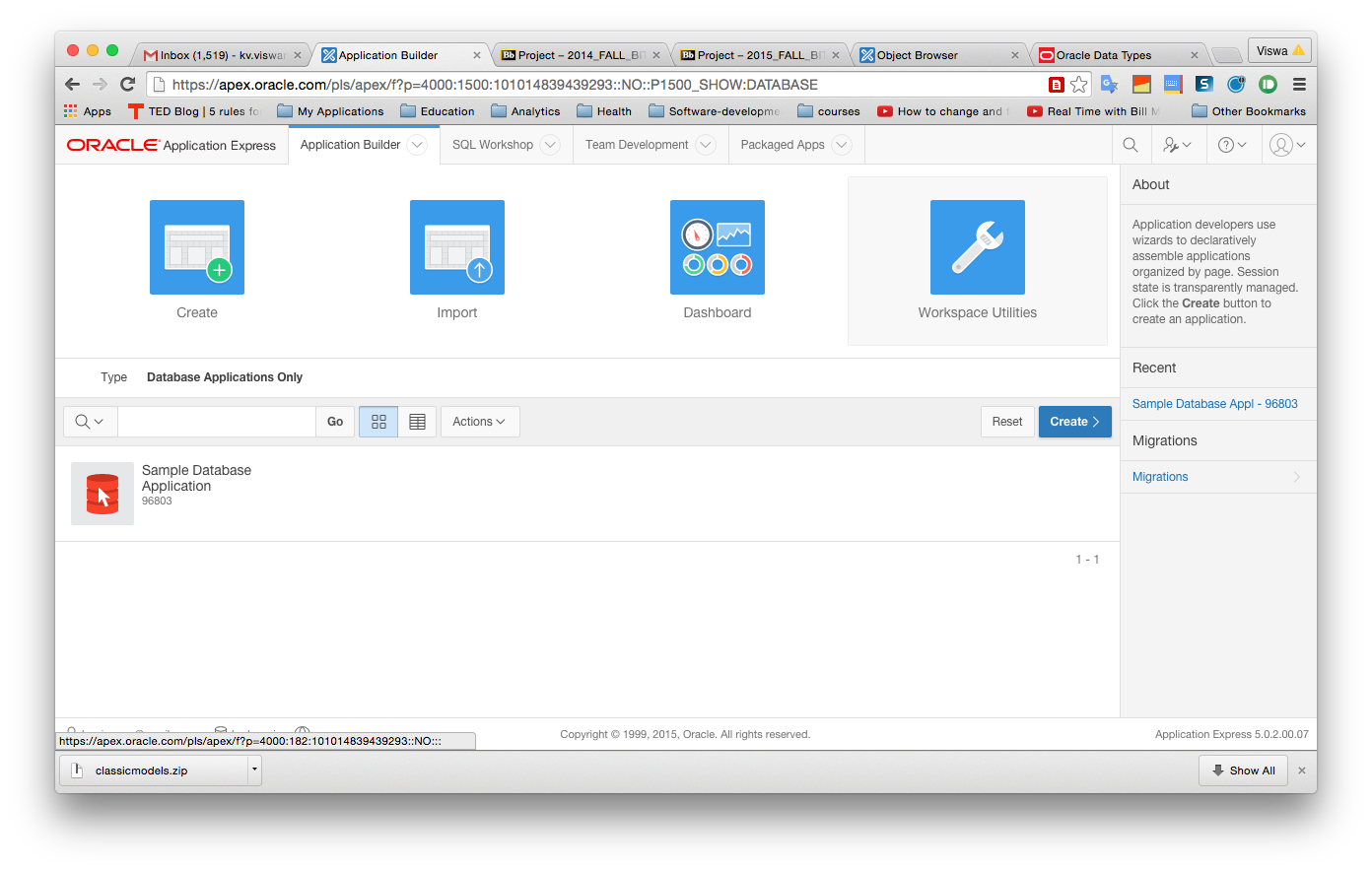
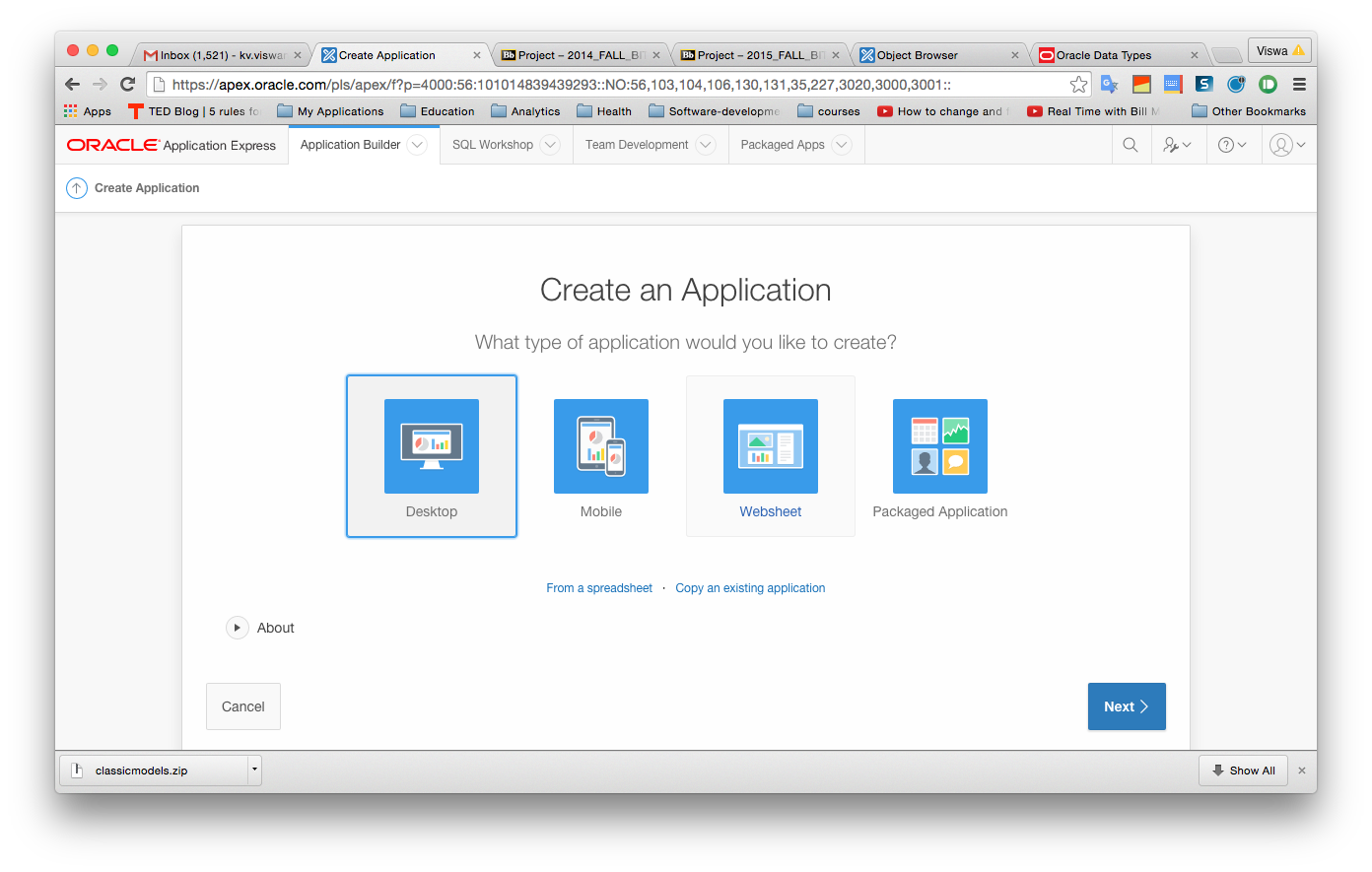


Figure : Listing of available applications

1. Click the *Create* button to create a new application. You will see the screen shown in Figure 11. Desktop should already be selected. Click "Next".



1. Figure : Creating a new application
2. On the next screen (Figure 12), give a name to your application and click "Next".

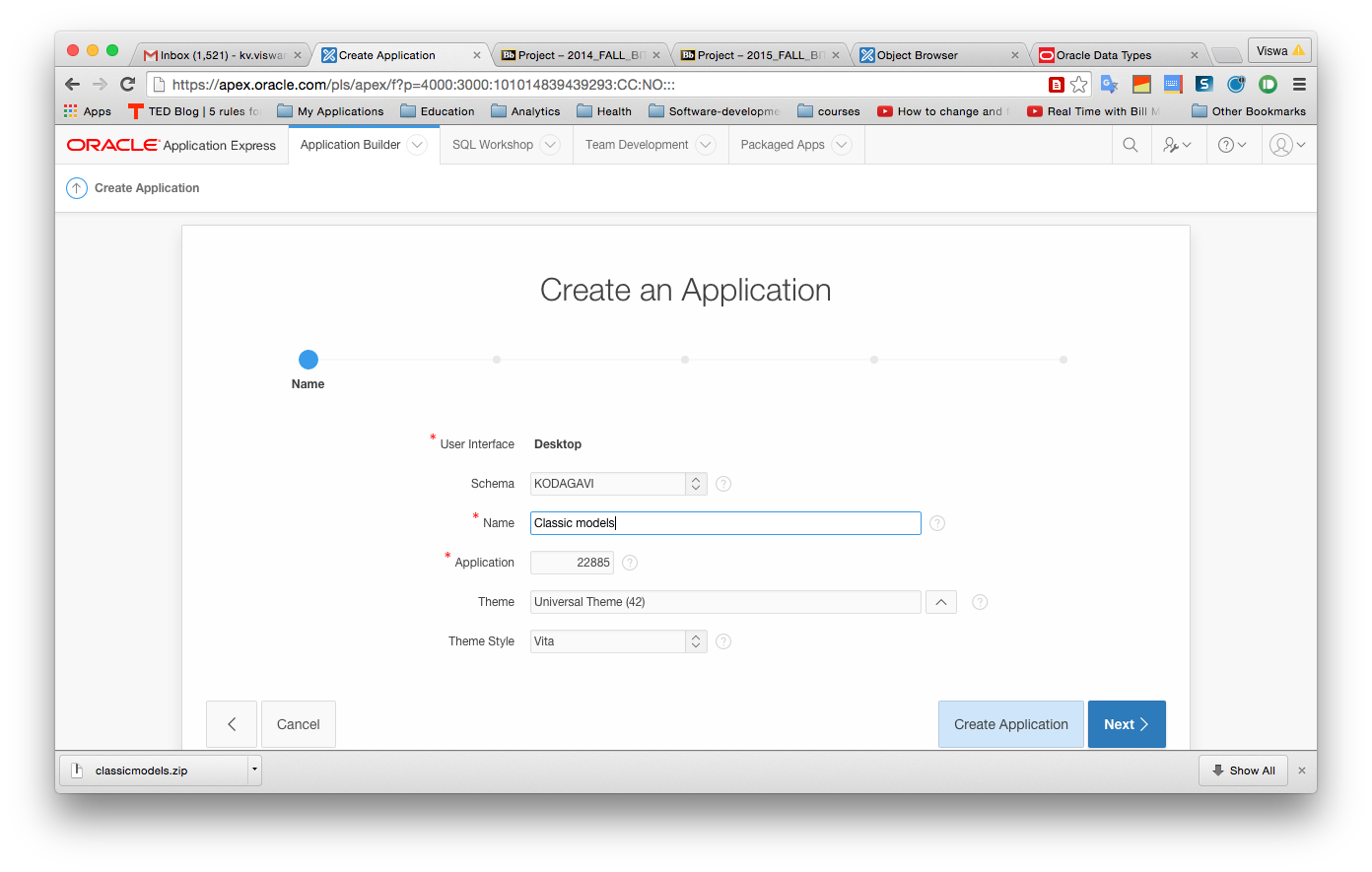


Figure : Naming your application

Leave the *Application\** field alone. In the *Name* field, enter a name for the application – like *Classic Models* (you can choose any name). Leave the *Schema* field alone. Click *Next*. On the next page, shown in Figure 13 below, we see that our application has one blank page. The screen has an option for us to add more pages to our application, but we will not add any more pages just yet.

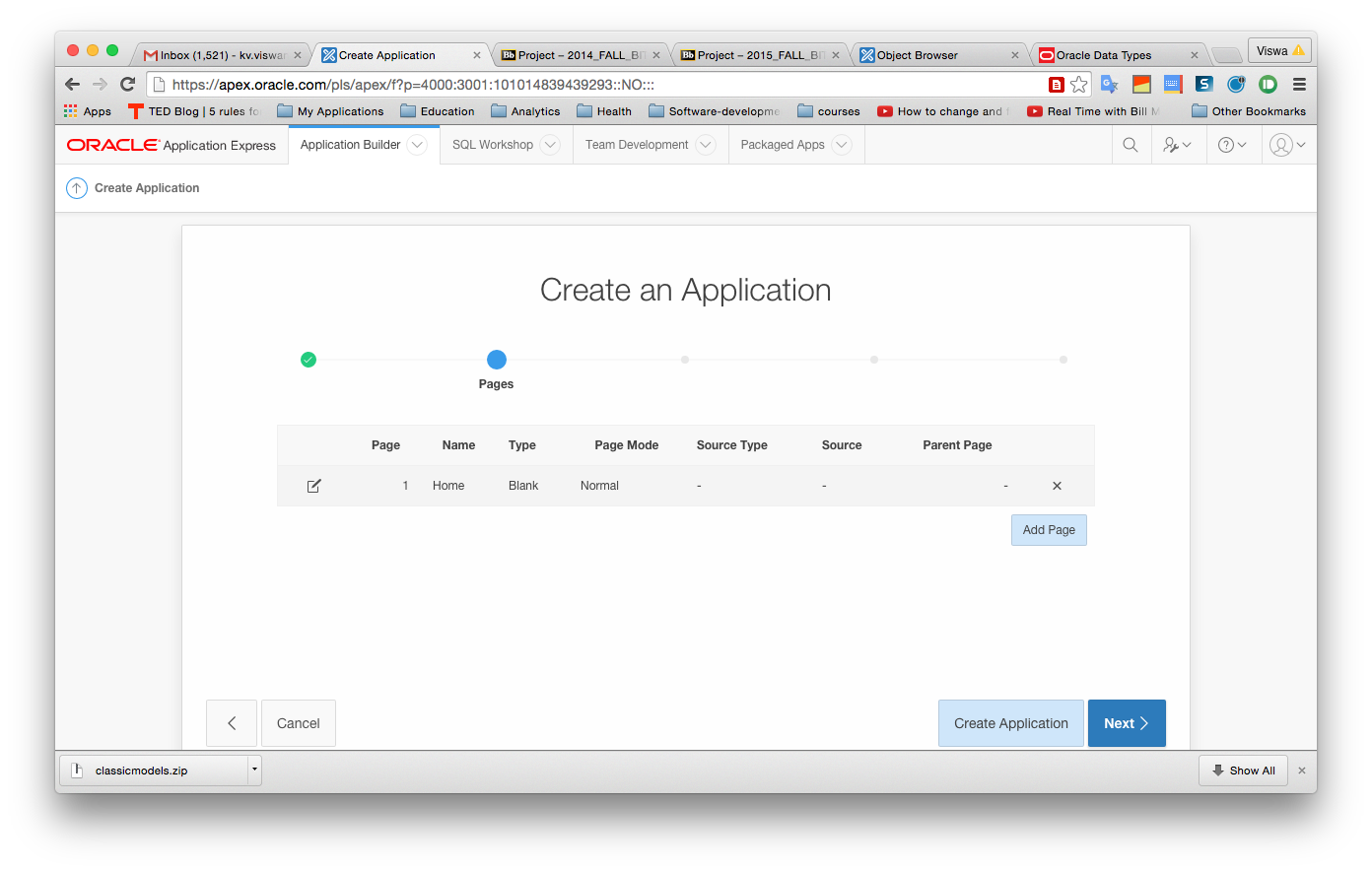


Figure : Our app has one page so far

1. Click on *Next.* On the next screen for selecting shared components leave the default as *No* and click *Next* again. The next screen looks like that shown in Figure 14 below. We select some important attributes of the application on this screen. Accept the defaults and click "Next" and on the next screen click "Create application".

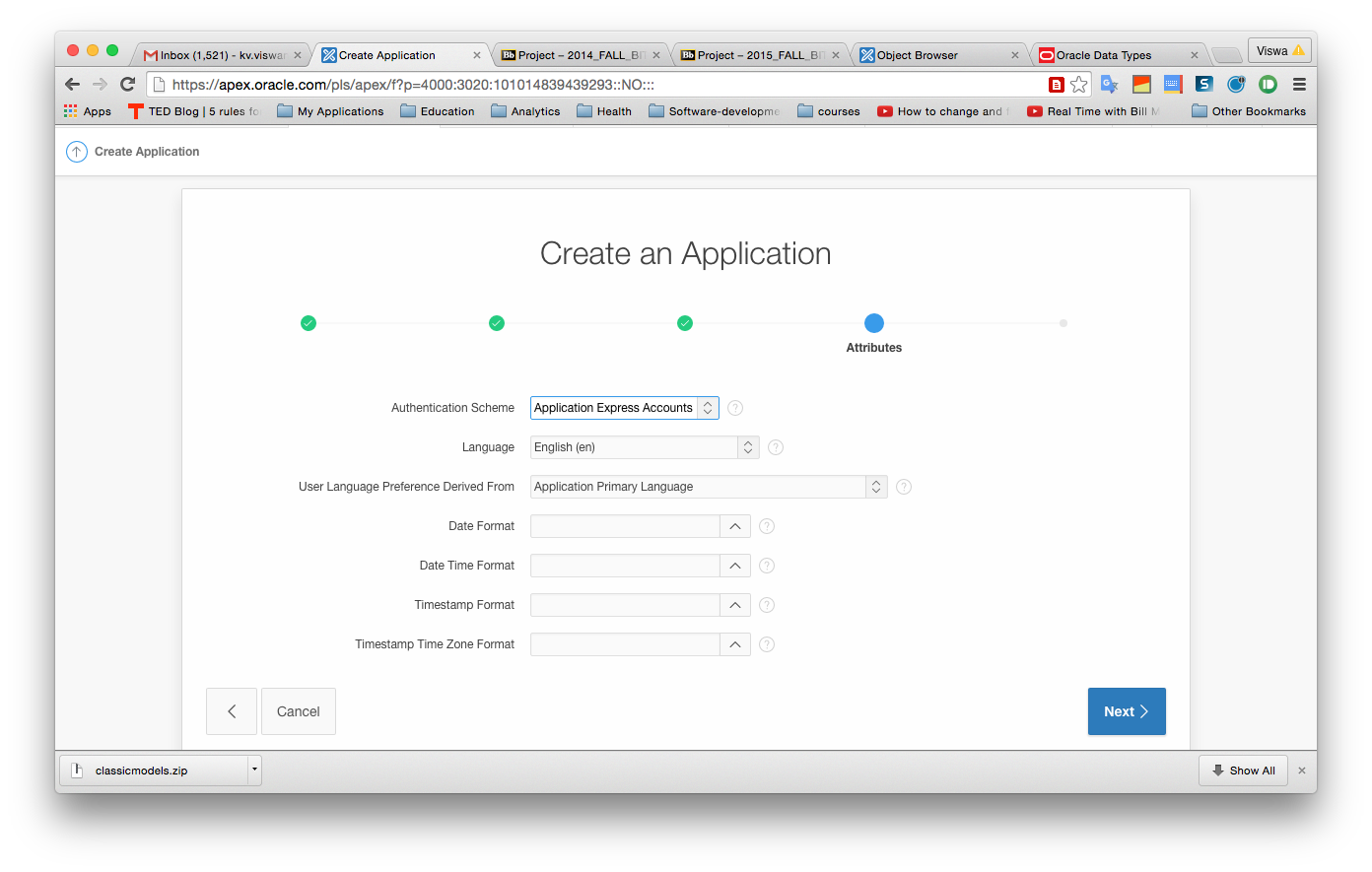
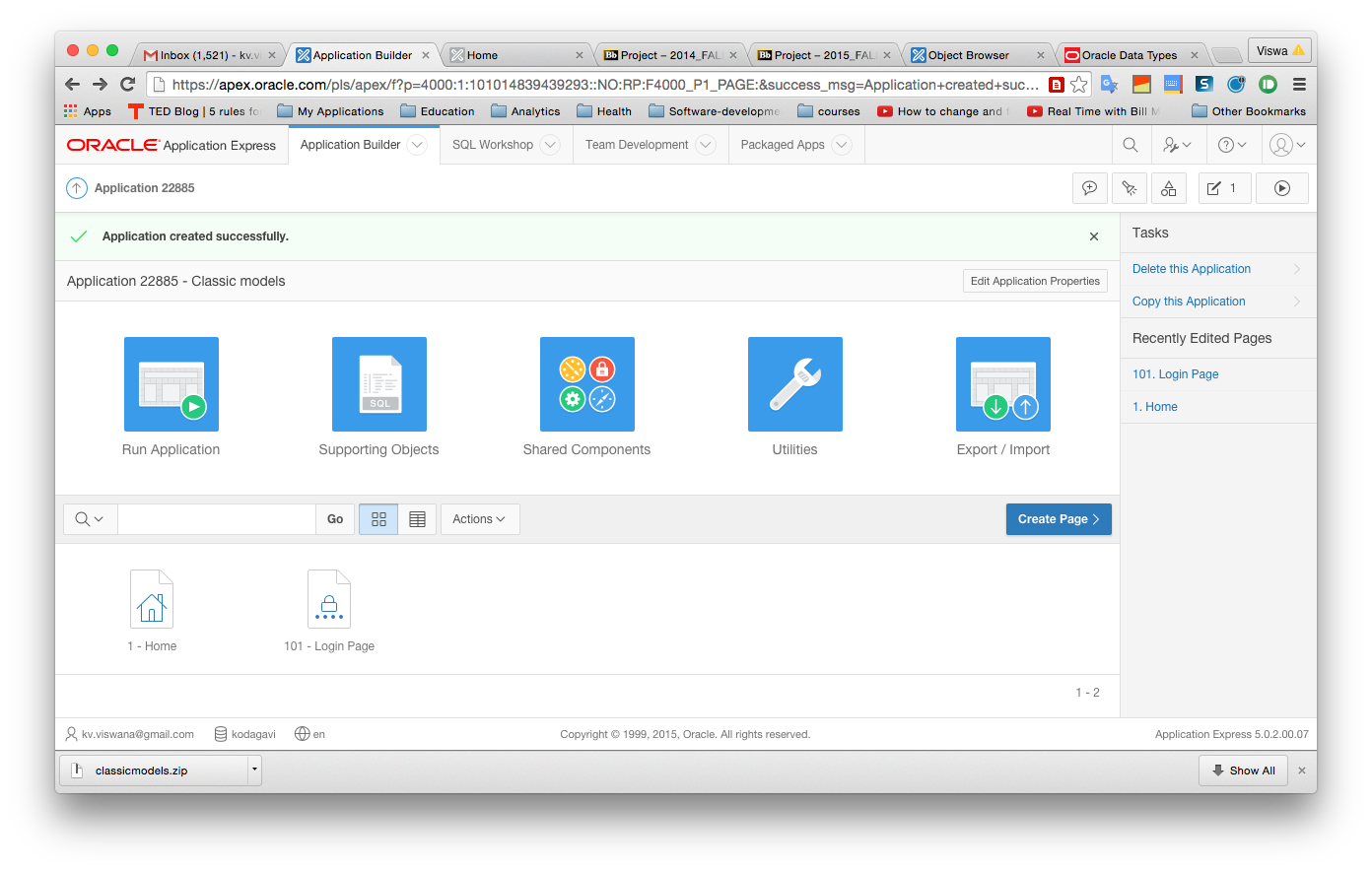


Figure : More application options

1. You now see that our application has two screens or pages – a login page and a blank page (Figure 15).



Application now has blank home page and a login page

Figure : Application with login page and a blank page

1. Your application shell is now ready. You can run it by clicking "Run". Of course, other than logging in, we cannot do much because we have not yet created any screens!
2. Nothing interesting, but we have a running application. Let us now add a page for managing customers to our fledgling application. At the bottom of the page you see several developer links. Click on *Application xxx* to return to developer mode to continue building the application. Click on *Create page*. On the next page, clock on *Form* (because we will need a form to edit or add customers). This takes you to the next page.
3. On the next page, ***Form on a table with report*** should already be already selected – just click "Next". We select this because you will recall that we first saw a report of all customers (if there were any) and then were able to access a form to create, edit or update a customer. If for some reason ***Form on a table with report*** is not selected, select it and you will be taken to the next page. *After some time,* you see the page shown in Figure 16. Fill in the page name as "Customer report" – because that is what we will be creating. Just click in the field "Region title" and it will automatically update to the page name you chose (after a brief delay – be patient!). Also select "Breadcrumb" in the last dropdown box. Then click "Next".

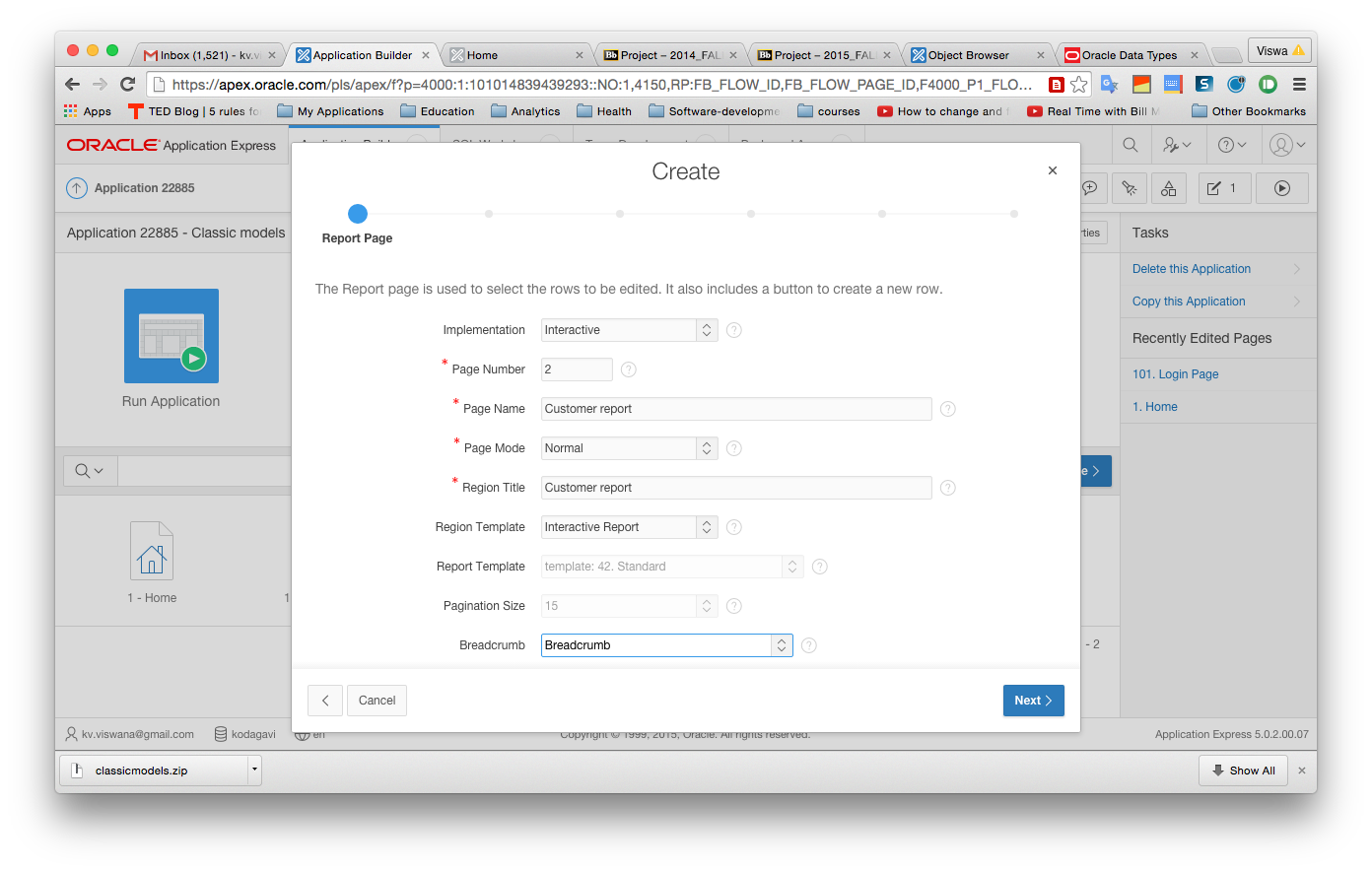


Figure : Form options

1. On this page we need to tell the system that the data for this report and associated form are going to come from the Customer table. Leave the *Table/View owner* alone. Select P\_*Customer* for *Table/View name* (Figure 17).

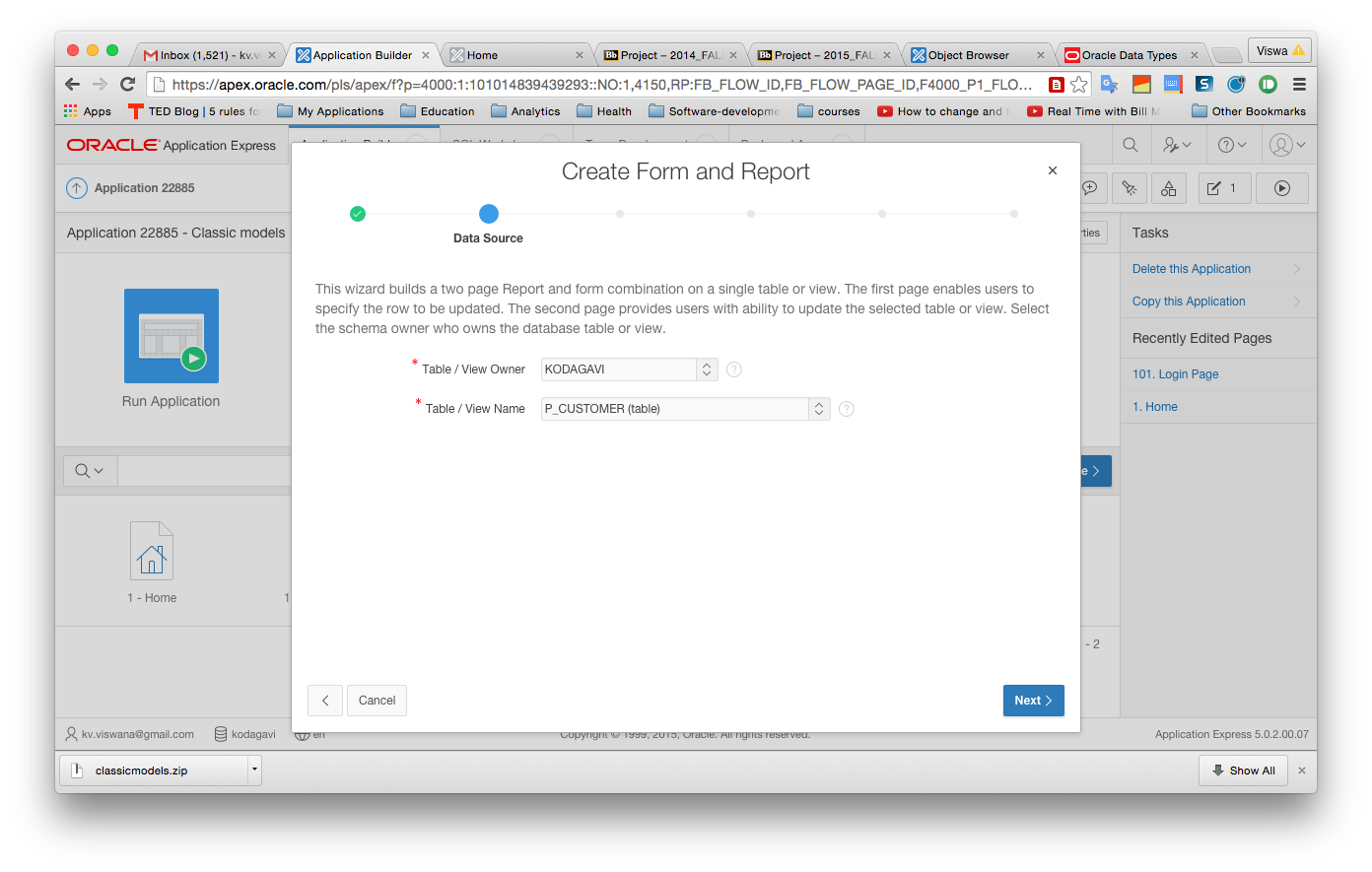


Figure : Selecting the table for the form

1. Click *Next*. This screen is asking about whether we want this report to be accessible from a tab and if so what should be the name on the tab. If you go back to the step where we first ran the application, you will note that we accessed the Customer report from a tab. So we do want this to be on a tab. Make the selections as shown in Figure 18. (Select "Create a new navigation entry" and select "Home" as the parent entry). Click "Next" after making these selections.

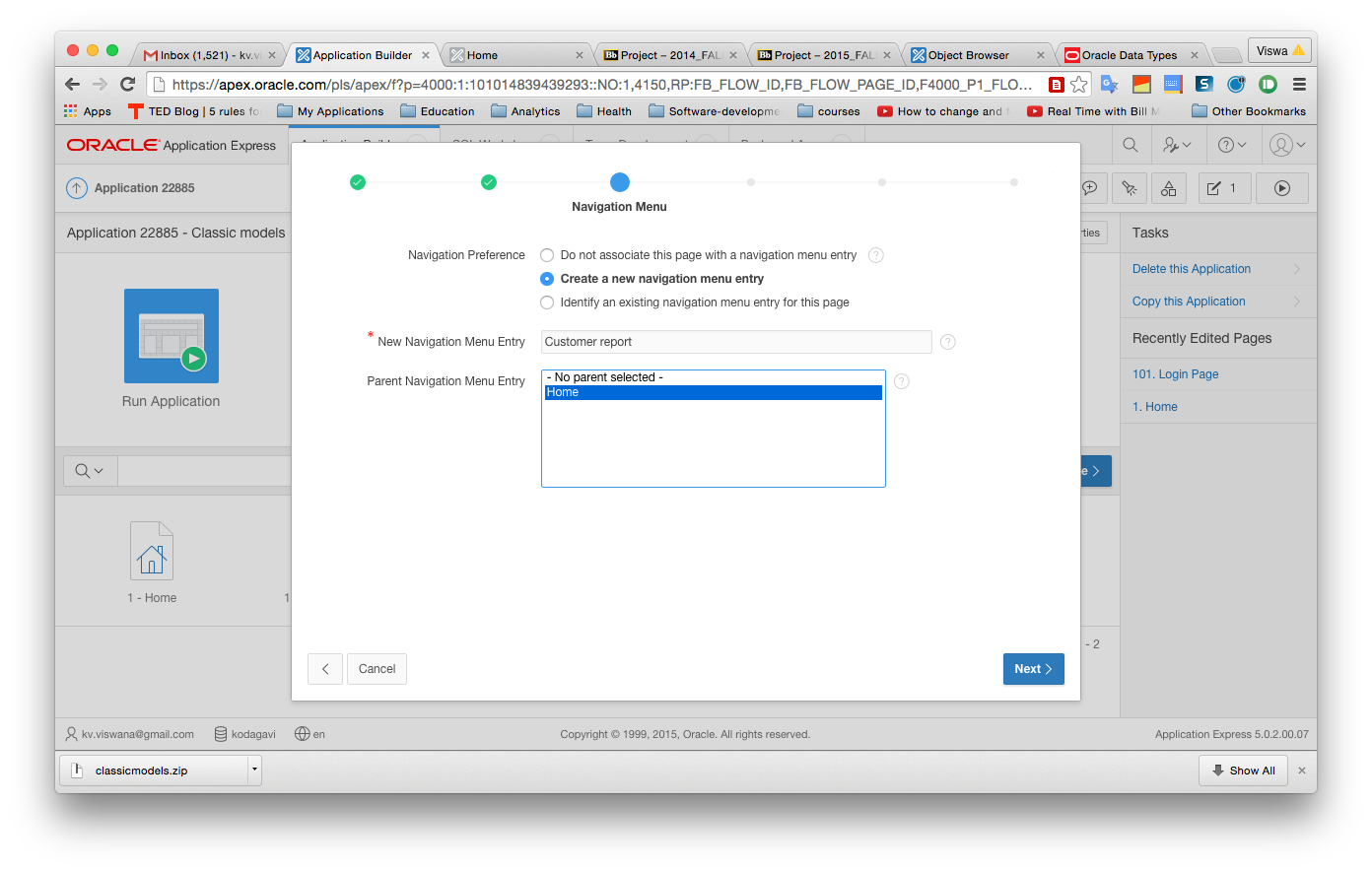


Figure : Setting up a tab for customers

1. The next screen (Figure 19) is asking about which columns of the customer table we want to display in the report. By default all the columns are selected (moved from left to right). Note from the screen that we can actually filter the rows by adding a WHERE clause. For now we will ignore that option and take all rows. Click "Next".

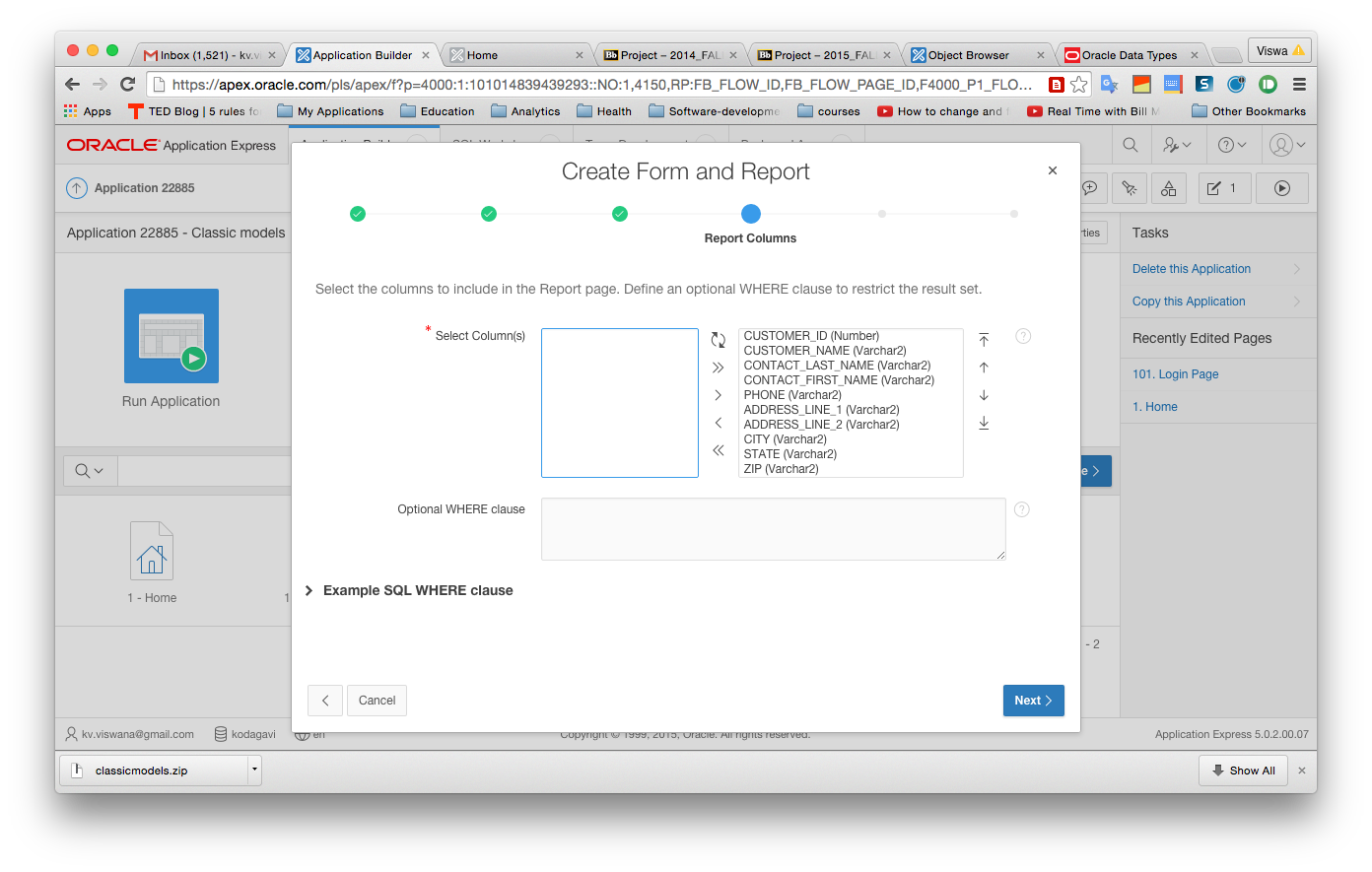


Figure : Selecting columns to display

1. When you finish the page application, you will be able to click on a report row to edit the row. This screen is asking what icon we want for the edit button. Pick any that you like. Click "Next".
2. The resulting screen is about the form that we will use to add or edit a customer – the form. What we earlier created was the report – a listing of customers. Change the page name to "Customer form". Then click inside the "Region title" field and it updates automatically. Then click "Next".
3. Now we get a screen asking about the primary key of the table. Fill it as shown below (Figure 20). and click "Next".

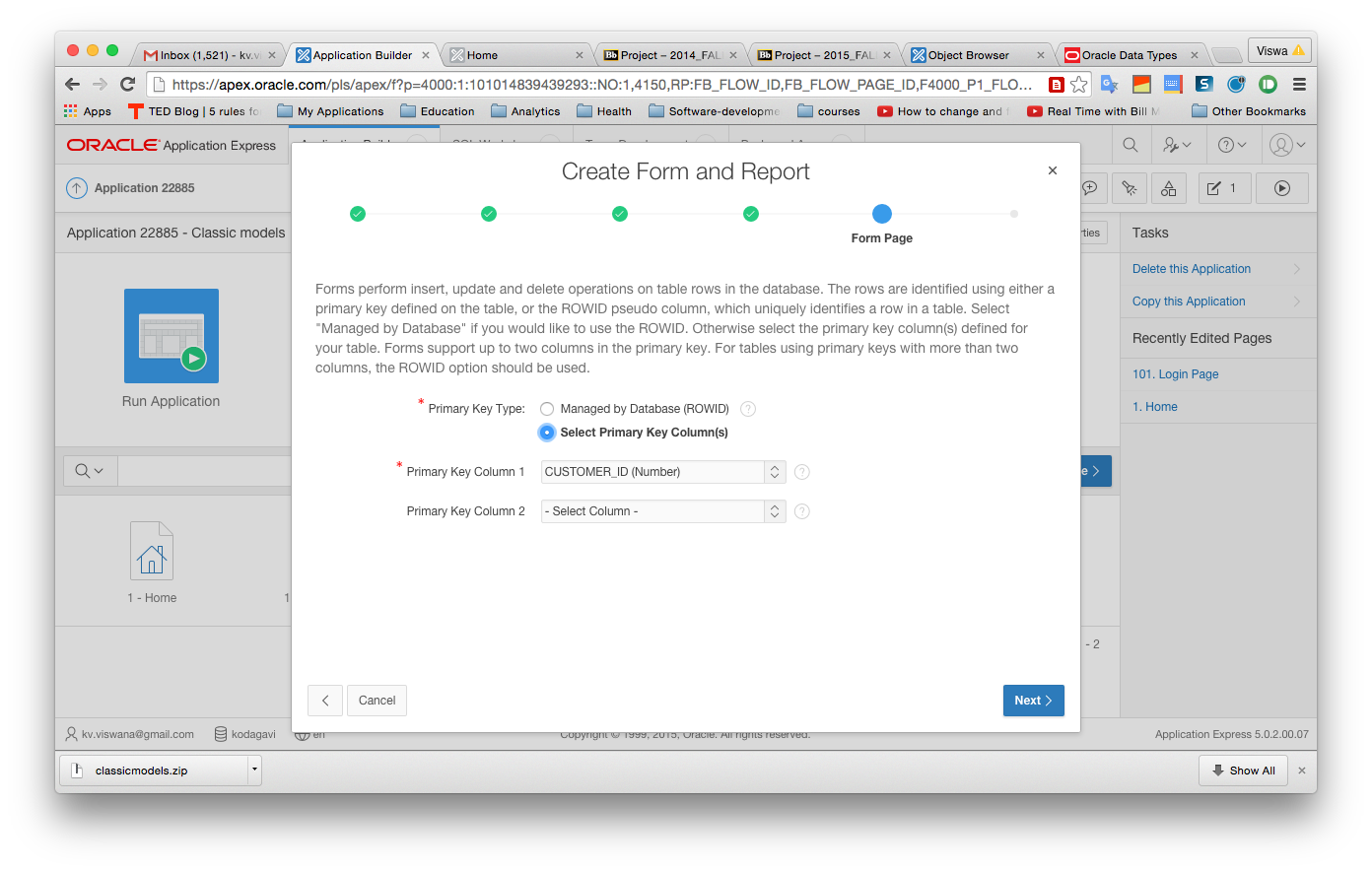


Figure : Selecting the primary key to use

1. The next screen is asking us about how we will be filling in values for customer\_id which is an integer primary key. As we have already seen we have set up an auto\_incrementing facility for this. Leave the screen as is (with "existing trigger" selected) and click *Next*.
2. The next screen wants us to specify what columns should appear on the form (we had earlier specified the columns for the report). Click on >> to select all columns. Then click "Next".
3. You get a screen that asks whether we want the form to be used for *insert*, *update* and *delete*. Click *Next*. Finally we get a screen that lists all our choices. Click *Create* on this screen*.* If all goes well, you should get a confirmation that the form has been created successfully.
4. You can now run the page alone by clicking on *Run page*. Rather than doing that, let us go back to the start and run the whole application. On the top you see the breadcrumb that says *Application builder -> Application xxxxx -> Page 2* (the xxxxx will be replaced with your application number and your page number might be different). Click on *Application xxxxx* and then choose the option to run it. Alternately, you can always click on the *Application Builder icon on top* and see the two applications. You can then run the *Classic models* application. You see the home screen, but now you also see the *Customer* *Report*. Click on it. Since we do not have any customers yet, the report is empty. Use the "Create" button to create a new customer.
5. Continue along the same vein to create similar reports for Employees, Product lines, Products and Offices only – for the others we will learn additional techniques in the next lab. You should do all of these to gain practice. As you do more, you will be able to breeze through this pretty quickly.
6. **Entering data** 
   1. First use the **employees report** and create **three new employees with made up data** – **for now, leave the manager\_id and office\_id** **blank for all employees**. In the next lab, we will see what to do about these. Once you add employees, you will see the employee\_id in the report view. This does not show in the form for editing an employee, because this is not an editable field.
   2. Once you have employees, you can add a few **customers**. Use some of the **employee ids that you already have for the employee\_id field on the customer form**.
   3. In similar vein, create at least one product line and a few products in it. Once again, create product lines first and then use the corresponding id in the product form.
   4. In the next lab on APEX, we will see how we can ease the process of entering foreign key values like employee\_id and product\_line id.
7. The next document on APEX shows you how to create a form for Sales orders, create Lists of Values and also how to spruce up the home page.