Chapter No. 4
"Rich Content and Mobile Integration"
In this package, you will find:
The author’s biography
A preview chapter from the book, Chapter no.4 "Rich Content and Mobile Integration"
A synopsis of the book’s content
Information on where to buy this book

About the Author

Hussein Nasser is an Esri award-winning senior GIS solution architect at Electricity and Water Authority, Bahrain. He is the author of two books on ArcGIS, Administering ArcGIS for Server and Learning ArcGIS Geodatabases, both published by Packt Publishing. In 2007, Hussein won the first place in the annual ArcGIS Server Code Challenge conducted at the Esri Developer Summit in Palm Springs, California, for using AJAX technology with ArcGIS for Server, which was not implemented back then. After his 8-year career as a GIS Architect in the leading Middle Eastern Engineering company, Khatib & Alami, where he spent time implementing various utility GIS systems based on Esri technology across the Middle East, Hussein decided to move to a more focused environment in Electricity and Water Authority back in Bahrain, his homeland. Here he can channel his expertise to develop a robust GIS utility solution that is fully integrated with the eGovernment project, which will help Bahrain march towards the smart grid. Beyond GIS, Hussein is fascinated by acute research topics; some of the papers he is currently working on are The Human API: A Software Interface to Prevent Cancer, Global Economic Crisis and Natural Disasters Quantum Detector, and Stock Market change with the Moon Phases.

I would like to thank Nada; most of this book was written in our favorite coffee shop. You wouldn't be holding this book if she wasn't there.

For More Information:
www.packtpub.com/web-development/building-web-applications-arcsdes
Building Web Applications with ArcGIS

Building Web Applications with ArcGIS is a short book. Short books are hard to write, because I have to condense essential information into less than 150 pages. It is challenging to determine what is essential when you know a lot about a particular subject. The writer has to sacrifice of some content so that they can produce a quality title that readers can really benefit from.

ArcGIS is a suite of software, developed by Esri—Environmental Systems Research Institute. ArcGIS allows its users to view, edit, analyze, and work with geographic data. You can work with geographic data on desktop, web, or mobile. This book tackles the web development side of ArcGIS; it teaches the reader how to build web applications that can interact with ArcGIS.

I am very proud of this title. It is a special book because I have tried a new writing style I haven't used before. This is the first book I have ever written that is purely based on a real-life project. As a reader, you act like a web development company where your clients hand you their requirements. Chapter by chapter you start building the application required by the client gradually: adding functionalities, studying their feasibility, and implementing accordingly. Not only will this teach you the basics of developments for ArcGIS, but it will also relate to your real-life projects as well.

I get bored when I read a book that is cluttered with methods and functions and I have to figure out when and where to use them. Some books give you examples disconnected from reality that you won't ever encounter in your lifetime. This book is different, as each method you use, each library you add, contributes to a requirement requested by a client and it makes sense. You will read and say "yes, this is something my client would definitely request".

Building Web Applications with ArcGIS was designed for web developers who don't necessarily have an experience of ArcGIS. There are going to be three themes running throughout the book. The first theme is design, which is covered in the first two chapters of this book. We will discuss how to interpret requirements, create the interface design, and add basic functionalities such as loading the map. The second theme is development, which is covered in Chapter 3, Querying ArcGIS Services. This is where the reader will add more functionalities such as querying and interacting with the map. The last theme is enrichment and is covered in Chapter 4, Rich Content and Mobile Integration and Chapter 5, Posting Reviews, Ratings, and Photos. It is designed for advanced readers. It will show how to do editing, querying related information, and mobile integration.

For More Information:  
www.packtpub.com/web-development/building-web-applications-arcgis
All the three themes come under the umbrella of a project called "Bestaurants", where the reader helps a client in Belize, a country on the northeastern coast of Central America. The reader will help improve the Bestaurants project by designing a web interface to visualize the best restaurants, diners, café, and so on in Belize. With each chapter, the Bestaurants' client will ask for new requirements, which the reader will try to implement by the end of the chapter.

What This Book Covers

Chapter 1, The Bestaurants Project, contains a full description of the Bestaurants project. It breaks down the requirements into small pieces that will be executed in the next four chapters. This chapter will also include some introduction about map services, JavaScript API, and how to set up the necessary web services.

Chapter 2, Setting Up the Basic Web Application, teaches you how to get started with a basic map web page based on the design proposed in Chapter 1, The Bestaurants Project. You will set up the web server, create a simple HTML page, and add necessary code to show the map service published in the previous chapter. You will be able to gradually, throughout the next chapters, fill the page with functionalities.

Chapter 3, Querying ArcGIS Services, teaches you to communicate with the services to query, retrieve, and display the results now that you have developed a basic web viewer website.

Chapter 4, Rich Content and Mobile Integration, makes the web application more interactive by adding more rich tools. You will query and display the related records and do some calculations with the results. This chapter will also enable our site to be viewed on mobile.

Chapter 5, Posting Reviews, Ratings, and Photos, introduces the feature service and editing. It will show you how your client can post restaurant reviews, ratings, and photos.

Appendix, Bestaurants on ArcGIS Online, discusses an alternative way to implement the web applications using ArcGIS online.

For More Information:  
www.packtpub.com/web-development/building-web-applications-arcgis
Our Bestaurants web application is coming in to shape. We have developed and added a lot of functionalities to the application; we started with a simple HTML skeleton, added the map, and interacted with the map with some tools and queries. We have reached the stage where we can add rich content to the application, content that is not as easily accessible. The restaurants layer contains some good attributes that we have used to display such as name, description, website, and rating. However, the reviews and pictures for each restaurant are stored in other tables.

We are half way through the completion of our project where we acquired solid skills in ArcGIS web development. In this chapter, we will dive deeper and learn how to query and fetch this related data and display them consistently in our application so that it becomes more appealing to the tourist users.

**Brief introduction to relationships**

We will start by learning about relationships. A relationship is the key property of any relational database that is where the relational database management system got its name from. ArcGIS uses a relational model which means we have to deal with relationships in our coding.
Rich Content and Mobile Integration

Take a moment and review the database model back in Chapter 1, The Bestaurants Project. In the Bestaurants scenario, our food_and_drinks layer is related to another table called VENUES_REVIEW. Each relationship has a unique identifier that we will use to query the related records. Open the food_and_drinks layer by visiting the link: http://arcgismachine:6080/arcgis/rest/services/Bestaurants/MapServer/0 and then scroll down to Relationships as you can see in the following screenshot:

Any relationship is composited of a primary key, which resides in the main or the source table. In our case it is food_and_drinks, and a foreign key that can be found in the destination table, Venues_reviews. The primary key of food_and_drinks is an autogenerated number by ArcGIS referred to as ObjectID, which is also the foreign key on the Venues_reviews table. So, we now know that to find the reviews for a restaurant feature f, we have to get the object ID of that f. We will learn how to do that in the following example.

The ObjectID is a numeric number, which uniquely identifies any ArcGIS table or class and functions as a primary key for this table.

So how do I see the fields in the VENUES_REVIEW table? You can guess that it will have a unique URL just like the food_and_drinks table. Click on VENUES_REVIEW (2) to open the review table or simply open the page. The VENUES_REVIEW table has an ID of 2 while the relationship between them has an ID of 0. It is important not to confuse the two. Take a look at the reviews table in the following screenshot:

For More Information:
www.packtpub.com/web-development/building-web-applications-arcgis
A simple relationship query example

Before diving into the actual implementation of our proposed interface design, which was to display the reviews in the results panel, we will work with a smaller example to understand how relationship queries work. In the following example, we will demonstrate how to get the review text and the name of the review user and display them in an alert message when the user clicks on a restaurant result:

1. You can continue from your previous mybestaurants.html file from Chapter 3, Querying ArcGIS Services. You can also get the same file from 2955OT_03_Files\Code\bestaurants07_queryhighlight, copy it to c:\inetpub\wwwroot, and rename it to mybestaurants.html.

2. First of all, the relationship query object that we will be using is located in a different library that we have to refer to. Add the following reference to your code:

   ```html
   ...
   <script>
   dojo.require("esri.map");
   dojo.require("esri.layers.featurelayer");
   dojo.require("esri.dijit.Legend");
   dojo.require("esri.tasks.query");
   dojo.require("esri.tasks.RelationshipQuery");
   function startup()
   {
   ...
   ...
   
3. Do you remember how we highlight a restaurant when a user clicks on the query result? We wrote a function called showRestaurant, and we will now go to that function and add some new code. We will first create a relationship query object, and then ask it to return all fields with "*" as done in the following code. Again, it is recommended to use the fields that you will absolutely require in your query for performance reasons. But, we will use "*" for simplicity:

   ```javascript
   //clear any graphics on the map
   map.graphics.clear();
   //so we only add this one
   map.graphics.add(record);
   //Create relationship query object
   var relatedReviews = new esri.tasks.RelationshipQuery();
   //return all fields in the related table.
   relatedReviews.outFields = ["*"];
   ```
4. We have to specify what the relationship ID is, which we saw in the previous screenshot. This is important for ArcGIS to be able to fetch that relationship metadata. Also, we have to give the ObjectID of the current clicked restaurant. Add the following code:

```javascript
//Create relationship query object.
var relatedReviews = new esri.tasks.RelationshipQuery();
//return all fields in the related table.
relatedReviews.outFields = ['*'];
//The relationship id is zero based on the url
relatedReviews.relationshipId = 0;
//get the object id
relatedReviews.objectIds = [record.attributes['OBJECTID']];
...
```

5. This code will definitely generate an error. The reason is that we never asked the query to return the ObjectID field, we only asked for the name and the rating. We have to add the ObjectID field to our query fields. Go back to the `executequery` function and modify the code accordingly as shown in the following code snippet:

```javascript
function executequery()
{
    q = document.getElementById('txtq').value;
    var query = new esri.tasks.Query();
    query.returnGeometry = true;
    query.outFields = ['OBJECTID', 'NAME', 'RATING'];
    query.text = q;
    queryTask.execute(query, showResults); 
}
...
```

6. It is time to run our query. We always run a query on the feature layer by calling the `queryRelatedFeatures` function. We pass our relationship query object, and a function will be called when the query is completed. Let's call it `RelationQueryComplete`. Add the following code:

```javascript
//Create relationship query object
var relatedReviews = new esri.tasks.RelationshipQuery();
//return all fields in the related table.
relatedReviews.outFields = ['*'];
//The relationship id is zero based on the url
```
relatedReviews.relationshipId = 0;
//get the object id
relatedReviews.objectIds = [record.attributes['OBJECTID']];
//execute the query and then call the RelationQueryCompelete
lyr_foodanddrinks.queryRelatedFeatures
(relatedReviews, RelationQueryCompelete);
...

7. Of course, we have to write our RelationQueryCompelete function; the
   function gives us a key-value pair array. The key is the object ID of the
   feature and the value is the actual related record. This, in our case, is the
   review. So, we have to loop through the records to get it. There might be
   more than one review for the same restaurant and you can loop through
   them and get them all. However, we are only interested in the first related
   record, features[0]. The related record can be treated as a feature without a
   geometry, still you can access the attributes property to get your attribute.
   We are interested in the REVIEW and USER fields. Add the following function
   to your code:
   ...
   functionRelationQueryCompelete(relatedRecords)
   {
      for (var oid in relatedRecords)
      {
         var fset = relatedRecords[oid];
         //get the first related record
         var firstrecord = fset.features[0];
         var review = firstrecord.attributes['REVIEW'];
         var user = firstrecord.attributes['USER'];
         alert("By " + user + ": " + review);
      }
   }
   ...

For More Information:
www.packtpub.com/web-development/building-web-applications-arcgis
8. Save the mybestaurants.html file and run the application http://arcgismachine/mybestaurants.html. Do a search on Mercy and click on **Mercy's Bar**. You should get a pop up that says **By Walter White: Love this place!** as you can see in the following screenshot. This means our query has worked. You can find the updated file at \2955OT_04_Files\Code\bestaurants01_getrelationonclick.html. This code is good even when there are no related records; the code will not enter the loop, and eventually won't show the results.

![Image of the application interface]

**For More Information:**
www.packtpub.com/web-development/building-web-applications-arcgis
Working with Restaurants' rich content

We learned in the previous section to do a simple relation query when the user clicks on a result. It was quite simple, as it was only a single feature at a time. However, what is required from us is something different. We need to actually show these reviews as the result is displayed, hence we will be dealing with multiple features at the same time. Because of the asynchronous nature of our code, we cannot use the conventional method of querying and waiting for the result, because the code will be long executed and the last result will already be displayed before we can even get the related records of the first result.

Asynchronous code is a piece of code that gets called by an initiator and is executed separately. The initiator does not wait for that code to be completed before resuming the rest of the code segments.

We will learn how to fix this in the next section.

Displaying reviews

We have to make some changes in the ShowResults function. This is the function that shows the query results and this is where we need to query for related records. While displaying the query result, we add a div element placeholder named by object id of the current feature. This will be our first step, as this way we have a review placeholder for each feature. Follow these steps to start displaying the reviews:

1. Edit the mybestaurants.html file and add the following code to ShowResults:

   ```html
   //display the name
   resulthtml = resulthtml + "<b>Name:</b> <a href='#' onclick = 'showRestaurant(" + i + ")'>" + record.attributes['NAME'] + "</a><br>
   //display the rating
   resulthtml = resulthtml + "<b>Rating:</b> " + record.attributes['RATING'];
   //create a place holder for each review to be populated later
   resulthtml = resulthtml + "<div id = 'review" + record.attributes['OBJECTID'] + '"></div>";
   //new line
   resulthtml = resulthtml + "<br><br>";
   ```

For More Information:

www.packtpub.com/web-development/building-web-applications-arcgis
2. We have marked the created div with the object ID, so we can access it later by simply specifying "review" + OBJECTID. Next, just before the end of ShowResults, call the AddReviews function and pass the same results object that got passed to ShowResults to it so that we can use it. This doesn't exist yet but we are going to add it in the following code snippet:

```javascript
... //add the results to myresults
element.document.getElementById("myresults").innerHTML = resulthtml;
}
AddReviews(results);
} // end of ShowResults

3. Next, add the AddReviews function, get the results, and create the basic loop. Use the ShowResults for preparation to query the related records. We will loop again on each result and send a query to get the related records:

```javascript
function AddReviews(results) {
    var resultCount = results.features.length;
    for (var i = 0; i<resultCount; i++) {
        //for each single feature or record in the result
        var record = results.features[i];
        // Relationship query code goes here
    }
}
```

4. Next, we need to create a relationship query object; by now you know how to do it. Let the results go to RelationQueryComplete that we created at the beginning of the chapter:

```javascript
... //Create relationship query object
var relatedReviews = new esri.tasks.RelationshipQuery();
```

For More Information:
www.packtpub.com/web-development/building-web-applications-arcgis
//return all fields in the related
table.relatedReviews.outFields = ["*"];

//The relationship id is zero based on the url
relatedReviews.relationshipId = 0;

//get the object id
relatedReviews.objectIds = [record.attributes["OBJECTID"]];  

//execute the query and wait, the RelationQueryComplete function will be called with the results once the query is finished
lyr_foodanddrinks.queryRelatedFeatures(relatedReviews, RelationQueryComplete);


5. The final change is in the RelationQueryCompelete function. We don't want to bug the user with alert messages, so instead we will simply update the corresponding review div object ID HTML tag. Fortunately, we have the object ID (oid) by looping through the key value pair in relatedRecords:

```
fuctionRelationQueryCompelete(relatedRecords)
{
    for (var oid in relatedRecords)
    {
        var fset = relatedRecords[oid];
        var firstrecord = fset.features[0];
        var review = firstrecord.attributes["REVIEW"];  
        var user = firstrecord.attributes["USER"];
        //update the div id with the review and the user
document.getElementById("review" + oid).innerHTML = "By " + user + ": " + review;
    }
}
```

6. Let's style the review: make both the review and the user italic.

```
//update the div id with the review and the user
document.getElementById("review" + oid).innerHTML = 
  
"<i>" + review + "<br>-" + user + "</i>"
```
That's it. Save your file and run it [http://arcgismachine/mybestaurants.html](http://arcgismachine/mybestaurants.html). You should see the reviews show up as you run the query. Run a query on Mercy as shown in the following screenshot:

![Querying Mercy Restaurant](image)

You can find the final code at 2955OT_04_Files\Code\bestaurnts02_queryreviews.html for displaying reviews.

Displaying pictures

Pictures of the restaurants are stored as attachments; the attachments are, in a way, related records; however, they are accessed in a much simpler method. Using the feature layer, we have to call the `queryAttachmentInfos` function, give it the object ID we want to retrieve the attachments for, and then parse the result in a separated function. Since it is an asynchronous call, we will use the same approach as the reviews; we will reserve some `div` elements for the restaurant pictures and display them after the result's load is completed.

The results will give us an attachment array. Each attachment has a `url` property that we are interested in to show a picture of that restaurant in the result. You can learn more about ArcGIS attachments at [http://qr.net/esriatt](http://qr.net/esriatt). Follow these steps to start displaying the pictures:

1. Edit `mybestaurants.html`, go to the `showResults` function, and add a place holder `div` element for the pictures as we did to the reviews using the following code:

   ```javascript
   //display the name
   resulthtml = resulthtml + "<b>Name:</b> " + 
   `<a href='#' onclick = 'showRestaurant(" + i + ")'>" + 
   record.attributes["NAME"] + 
   "</a><br>
   //display the rating
   resulthtml = resulthtml + "<b>Rating:</b> " + 
   record.attributes["RATING"]; 
   //create a place holder for each review to be populated later
   resulthtml = resulthtml + "<div id = 'review" + 
   record.attributes["OBJECTID"] + 
   "></div>";
   //create a place holder for each attachment picture to be
   populated later
   resulthtml = resulthtml + "<div id = 'picture" + 
   record.attributes["OBJECTID"] + 
   "></div>";
   
   //display the name
   result.html = resulthtml + "<b>Name:</b> " + 
   `<a href='#' onclick = 'showRestaurant(" + i + ")'>" + 
   record.attributes["NAME"] + 
   "</a><br>
   //display the rating
   result.html = resulthtml + "<b>Rating:</b> " + 
   record.attributes["RATING"]; 
   //create a place holder for each review to be populated later
   result.html = resulthtml + "<div id = 'review" + 
   record.attributes["OBJECTID"] + 
   "></div>";
   //create a place holder for each attachment picture to be
   populated later
   result.html = resulthtml + "<div id = 'picture" + 
   record.attributes["OBJECTID"] + 
   "></div>";
   ```

2. We will need to add a call to a new `AddPictures` function right after `AddReviews` passes the `results` query because we are going to use it:

   ```javascript
   document.getElementById("myresults").innerHTML = 
   resulthtml;
   }
   AddReviews(results);
   AddPictures(results);
   ```
3. Next, we create the `AddPictures` function as follows. This looks similar to the `AddReviews` function. Loop through the results and prepare the function to query the attachments. There is only one line to be added: the `queryAttachmentInfos` function. When the attachment query completes, we want it to call the `PicturesQueryComplete` function that we will add next:

```javascript
function AddPictures(results) {
    var resultCount = results.features.length;
    for (var i = 0; i < resultCount; i++) {
        // for each single feature or record in the result
        var record = results.features[i];
        // Attachment code goes here
        lyr_foodanddrinks.queryAttachmentInfos([record.attributes["OBJECTID"]], PicturesQueryComplete);
    }
}
```

4. Finally, we write `PicturesQueryComplete`. This will receive an array of pictures. We are only interested in the first picture. We will pick it up and create an `img` HTML element and place it in the corresponding `div` element by its object ID as shown in the following code. Note that we have used the `URL` property in the attachment to show the picture. We have resized the picture's width and height to scale it properly in the results panel:

```javascript
function PicturesQueryComplete(pictures) {
    // in case of no pictures quit.
    if (pictures.length == 0) return;

    // get the first picture
    var p = pictures[0];

    // set it to its place holder
    document.getElementById("picture" + p.objectId).innerHTML = "<img src='" + p.url + '" width='200px' height='130px'/>";
}
```
Save your file and test it by querying Fern Diner. Your application should now display pictures of the restaurants. Take a look at our latest work in the following screenshot:

You may find the latest code at 2955OT_04_Files\Code\bestaurants03_pictures.html.

For More Information:
www.packtpub.com/web-development/building-web-applications-arcgis
Mobile integration

Smart phones are becoming a gadget that almost every person has. This has caused businesses to work harder to make their applications and technologies run on mobile. Having the GPS integrated in mobile devices has also boosted the need for Geographic Information Systems (GIS) applications for mobiles. This is why mobile integration of applications is crucial especially with our Bestaurants project, since tourists will be browsing Bestaurants from their mobile most of the time.

The fact that we designed and coded our ArcGIS Web application in JavaScript API makes it easier for us to view it on mobile devices. JavaScript is supported on mobile devices and can be rendered nicely on small handheld devices.

Testing the website on mobile devices

It is important to note that this application was developed to be accessed by both web and mobile users. We will add a small setting that allows us to access the application using mobile devices. To test our website on mobile devices, first, both our web server and the mobile device should connect to the same network. Second, we will need the IP address instead of the machine name. Of course, we will need to add one small tag in our HTML page that will allow our application to be viewed and scaled nicely on mobile devices. Follow these steps to allow running your application on mobile devices:

1. Edit mybestaurants.html and add the viewport tag at the beginning of the page. This will help render the page in mobile devices:

   `<html>
   <head>
   <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no"/>
   <title>Bestaurants Web Application</title>
   <link rel="stylesheet" href="arcgisjs/esri.css">
   <script src="arcgisjs/3.10/init.js"></script>
   </head>
   </html>`

2. Save mybestaurants.html and close it.

3. Get the IP address of your machine by typing the IPconfig command in the command prompt. Mine is 192.168.1.2.

4. We have to enable the Windows Firewall to allow access to port 6080. This is the port our services are running on. Type `wfc.msc` in the run dialog box to open the Windows Firewall options.

For More Information:

www.packtpub.com/web-development/building-web-applications-arcgis
5. Right-click on **Inbound Rule** and select **New Role**.

6. In the **Rule Type** dialog box, select **Port**, and click on **Next**.

7. Go to **TCP | Specific Local Port** and then enter 6080 and click on **Next** as shown in the following screenshot:

   ![Protocol and Ports screenshot](image)

   - **TCP**
   - **UDP**
   - **Specific local ports:**
     
     Example: 80, 443, 5000-5010

8. Select **All the connection** and click on **Next**.

9. Keep all the options checked (**Domain | Private and Public**) and click on **Next**.

10. Type **ArcGIS for Server Port** in the name and create the rule by clicking on **Finish**.

11. Now that the connections are open, we can safely browse our web application. Open the browser on your mobile device and type in the link **http://192.168.1.2/mybestaurants.html**. Make sure you are connected to the same Wi-Fi network your **arcgismachine** is connected to (**http://192.168.1.2/mybestaurants.html**).

You should see your application and be able to browse it similar to the Web.

**GPS integration**

By using the satellites, the **Global Position System (GPS)** allows a device's location to be identified. Most mobile devices have a GPS device integrated inside them. Using GPS, users can define their unique locations throughout the world with few meters resolution and can define their (latitude, and longitude) information. We can add some small script in our Bestaurants application to get the latitude and longitude of the device that is currently browsing the application. We will use the built-in navigator object in the browser to achieve that.

---

For More Information:

Follow these steps to add the GPS integration:

1. Edit the mybestaurants.html file and add the following code before the `startup` function:

```javascript
//Get GPS, save them in these variables for later use
var currentlat;
var currentlong;
if (navigator.geolocation)
{
    navigator.geolocation.getCurrentPosition(showPosition);
}
else
{
    alert( "Geolocation is not supported by this browser."
)
}

function showPosition(position)
{
    //save the position for later use
    currentlat = position.coords.latitude;
    currentlong = position.coords.longitude;
}
```

2. Save and run the code. You will be prompted with the following message by the browser. Click on **Allow** to allow the navigator to find your location. If you are running the application from a computer, the navigator will use the Wi-Fi network to determine the nearest location, which is not very accurate. If you are running the application from a mobile, the navigator will use the GPS device in your mobile to find the exact location. In short, your location is approximated by using either GPS, which uses satellites and knows your location within a few meters, Wi-Fi, which uses the location of the nearby Wi-Fi networks, or the cell tower, which uses the connection to a cellular network that can be accurate up to a few thousand meters. The location can also be improved with an accelerometer, compass, gyroscope, or barometer in your phone. This was taken from the official Google support (https://support.google.com/gmm/answer/2839911?hl=en).

For More Information:
www.packtpub.com/web-development/building-web-applications-arcs
3. Since the GPS coordinates is a point, and we have added a point element before, use the same code or just type in the following function named `showGPSLocation`. Note that I have hardcoded fake coordinates to show the coordinates. The reason is that unless you are in Belize, you cannot actually see this point. The GPS location will basically add a point to your current location:

```javascript
function showGPSLocation()
{
    // this is a fake coordinates, since you will not be in Belize
    uncomment this if you are in fact in Belize
    currentlat = -88.21;
    currentlong = 17.50;
    var symbol;
    // mark up symbol are for points.
    symbol = new esri.symbol.SimpleMarkerSymbol();
    // set the size
    symbol.setSize(50);
    // and the color (purple) and transparency of 50%
    (0.5) symbol.setColor(new dojo.Color([255, 0, 255, 0.5]));
    // create a graphic object
    var graphic = new esri.Graphic
    {
        // Point coordinates are the gps coordinates, create a point
        new esri.geometry.Point(currentlat, currentlong, map. spatialReference), // the symbol of the point
        symbol
    };

    map.graphics.clear();
    map.graphics.add(graphic);
}
```

4. Finally, we will add a button to show the GPS location in our toolbar that calls the `showGPSLocation` function as shown in the following code snippet:

```html
Restaurant Name <input type = 'text' id = 'txtq'>
<input type = 'button' value = 'search' onclick = 'executequery()'>
<input type = 'button' value = 'GPS' onclick = 'showGPSLocation()'>
</td>
</tr>
```

For More Information:
www.packtpub.com/web-development/building-web-applications-arcgis
5. Save and run mybestaurants.html and search for Faber. You should get the following screen as the final result:

You can find the final application code with the GPS integration at 2955OT_04_Files\Code\bestaurants04_withGPS.html.

**Summary**

In this chapter, we injected the Bestaurants web application with rich features. We learned how to perform relationship queries and used them to view the names and reviews of the people who reviewed the restaurants. We then used the same approach to fetch the pictures of the restaurants that were stored as attachments in the feature layer and displayed them on the results. This made the web application more friendly and appealing to the users. We have finally ported the application fully to the mobile by adding the necessary code and enabling the ArcGIS for Server port so that mobile devices can use the service over the Wi-Fi network.
In the next and final chapter, you will learn how to edit feature layers to allow the users to add their own reviews and upload pictures. Editing requires special license of ArcGIS for Server in addition with an enterprise geodatabase. It can also be achieved with ArcGIS Online as we will see in Appendix, Restaurants on ArcGIS Online. Before moving to the next chapter, make sure to grab my book Learning ArcGIS Geodatabases, published by Packt Publishing, and read the last chapter to learn how to fully set up and configure an enterprise geodatabase using Microsoft SQL Server Express.

For More Information:
www.packtpub.com/web-development/building-web-applications-arcgis
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.