Data Access

By: Mosh Hamedani

Strategies

- **Application Properties:** for storing simple values (e.g. application settings, transient data)
- **File System:** for storing files (e.g. HTML, image, etc)
- **SQLite**: for storing structured data in a relational database
- **RESTful Services:** for storing data in the cloud

Application Properties

```
public class App : Application
{
     private const string TitleKey = “Title”;

     public string Title
     {
         get
         {
              if (Properties.ContainsKey(TitleKey))
                 return Properties[TitleKey];

              return “”;
         }
         set { Properties[TitleKey] = value; }
}
```

Data Access

By: Mosh Hamedani

File System

Use **PCLStorage** Nuget package.

https://github.com/dsplaisted/PCLStorage

SQLite

Decorate your domain objects with attributes:

- Table(“Recipes”): to override the table name
- Column(“RecipeId”): to override the column name
- PrimaryKey
- AutoIncrement
- Ignore
- Unique

```
public class Recipe
{
     [PrimaryKey, AutoIncrement]
     public int Id { get; set; }

     [MaxLength(255)]
     public string Name { get; set; }
}
```

Data Access

By: Mosh Hamedani

Get a **SQLiteAsyncConnection** object and then create a table.

```
var conn = DependencyService.Get<ISQLiteDb>().GetConnection();
await conn.CreateTable<Recipe>();
```

CRUD operations:

```
await conn.InsertAsync(obj);
await conn.UpdateAsync(obj);
await conn.DeleteAsync(obj);

var recipes = await conn.Table<Recipe>().ToListAsync();
var recipes = await conn.Table<Recipe>().Where(…).ToListAsync();


var recipe = await conn.Get<Recipe>(1);
```

To learn more:

https://github.com/praeclarum/sqlite-net

Data Access

By: Mosh Hamedani

INotifyPropertyChanged

To notify UI about changes in the state of an object:

```
public class Recipe : INotifyPropertyChanged
{
     private string _name;

     public string Name
     {
          get { return _name; }

        set
        {
            if (_name == value)
               return;

            _name = value;

            OnPropertyChanged();
        }
     }

     private void OnPropertyChanged([CallerMemberName]
                propertyName = null)
     {
         PropertyChanged?.Invoke(this, new
             PropertyChangedEventArgs(propertyName);
     }
}
```

Data Access

By: Mosh Hamedani

Consuming Web Services

Add **Microsoft.Net.Http** and **Newtonsoft.Json** Nuget packages to all projects.

```
var client = new HttpClient();


// Get resources
var content = client.GetStringAsync(url);
var posts = JsonConvert.DeserializeObject<List<Post>>();



// Create a resource
var post = new Post();
var content = JsonConvert.SerializeObject(post);
client.PostAsync(url, new StringContent(content));


// Update a resource
var content = JsonConvert.SerializeObject(post);
client.PutAsync(url + "/" + post.Id, new StringContent(content));



// Delete a resource
client.DeleteAsync(url + "/" + post.Id);
```