SharedPreferences ile Kullanıcı Adı Kaydetme Projesi

Erciyes Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği

Ders: Mobile Application Development

Öğretim Üyesi: Dr. Öğr. Üyesi Fehim KÖYLÜ

Proje Ödevi: SharedPreferences ile Kullanıcı Adını Kaydetme

İçindekiler

Giriş

Bu proje, Flutter ve Dart kullanarak mobil uygulamalarda yerel depolama işlemlerinin nasıl gerçekleştirilebileceğini göstermektedir. Özellikle, kullanıcı adlarının kalıcı olarak saklanması ve yönetilmesi için SharedPreferences API'sinin kullanımını içermektedir.

Mobil uygulamalar için yerel depolama çözümleri, kullanıcı tercihlerini saklamak, oturum bilgilerini yönetmek veya çevrimdışı verilerle çalışmak için önemlidir. Bu projede, uygulamanın basit bir gereksinimi olan kullanıcı adlarını kaydetme ve silme işlemlerini gerçekleştiren bir örnek sunulmuştur.

SharedPreferences Nedir?

SharedPreferences, mobil uygulamalarda anahtar-değer çiftlerini kalıcı olarak saklamak için kullanılan bir yerel depolama mekanizmasıdır. Flutter'da, `shared_preferences` paketi aracılığıyla bu işlevsellik sağlanır.

Gerçek uygulamalarda, bu paket kullanıcı tercihlerini, oturum bilgilerini ve diğer küçük veri parçalarını depolamak için idealdir. Bu paketi kullanarak şu veri türlerini saklayabilirsiniz:

- String (setString / getString)
- int (setInt / getInt)
- double (setDouble / getDouble)
- bool (setBool / getBool)
- List<String> (setStringList / getStringList)

Bu projede, test ve geliştirme amacıyla, gerçek SharedPreferences API'sini taklit eden bir `MockSharedPreferences` sınıfı kullanılmıştır.

Proje Yapısı

Proje, aşağıdaki ana bileşenlerden oluşmaktadır:

1. **MockSharedPreferences Sınıfı**: SharedPreferences API'sini taklit eden bir sınıf
2. **MyApp Sınıfı**: Uygulamanın ana giriş noktası ve tema ayarları
3. **HomePage Sınıfı**: Kullanıcı adlarını yönetmeye yönelik StatefulWidget
4. **_HomePageState Sınıfı**: HomePage'in durumunu yöneten ve tüm iş mantığını içeren sınıf

Kod Analizi

MockSharedPreferences Sınıfı

Gerçek bir SharedPreferences sınıfının davranışını taklit eden bu sınıf, projenin test edilebilirliğini artırmaktadır. Singleton tasarım desenini kullanarak, uygulamanın her yerinde aynı örneğin kullanılmasını sağlar.

```dart
class MockSharedPreferences {
  static final MockSharedPreferences _instance = MockSharedPreferences._internal();
  Map<String, dynamic> _data = {};

  factory MockSharedPreferences.getInstance() {
    return _instance;
  }

  MockSharedPreferences._internal();

  Future<bool> setString(String key, String value) async {
    _data[key] = value;
    return true;
  }

  String getString(String key) {
    return _data[key] as String? ?? "";
  }

  Future<bool> remove(String key) async {
    _data.remove(key);
    return true;
  }
}
```

Burada dikkat edilmesi gereken noktalar:

- `_instance` statik değişkeni ile Singleton örneği oluşturulur
- `getInstance()` metodu ile tek örneğe erişim sağlanır
- `setString()`, `getString()` ve `remove()` metotları SharedPreferences API'sini taklit eder
- Veriler bellekte bir Map yapısında tutulur

Kullanıcı Adı Kaydetme İşlemi

Kullanıcı adlarının kaydedilmesi için uygulama, JSON formatını kullanarak liste halinde veri saklar. Aşağıdaki metotlar bu işlemi gerçekleştirir:

dart

```dart
// Kaydedilmiş kullanıcı adlarını yükle
_loadSavedNames() async {
  final prefs = MockSharedPreferences.getInstance();
  setState(() {
    String namesJson = prefs.getString('userNames');
    if (namesJson.isNotEmpty) {
      try {
        List<dynamic> decoded = jsonDecode(namesJson);
        _userNames = decoded.cast<String>();
      } catch (e) {
        _userNames = [];
      }
    } else {
      _userNames = [];
    }
  });
}

// Kullanıcı adını kaydet
_saveName() async {
  final prefs = MockSharedPreferences.getInstance();
  final name = _nameController.text.trim();

  if (name.isNotEmpty) {
    setState(() {
      _userNames.add(name);
    });

    // Listeyi JSON olarak kaydet
    await prefs.setString('userNames', jsonEncode(_userNames));

    _nameController.clear();
    _showSnackBar("Kullanıcı adı başarıyla kaydedildi!");
  } else {
    _showSnackBar("Lütfen bir kullanıcı adı girin!");
  }
}
```

Burada:

1. `_loadSavedNames()` metodu, uygulama başlatıldığında daha önce kaydedilen kullanıcı adlarını yükler
2. Kaydedilen veriler JSON formatında tutulduğu için `jsonDecode()` ile çözümlenir
3. `_saveName()` metodu, yeni bir kullanıcı adı ekler ve güncellenmiş listeyi JSON olarak kaydeder

4. Boş ad girişleri kontrol edilir ve kullanıcıya geri bildirim verilir

Kullanıcı Adı Silme İşlemi

Uygulamada bireysel kullanıcı adlarını veya tüm listeyi silme işlevselliği de bulunmaktadır:

```dart
// Kaydedilen tüm kullanıcı adlarını sil
_clearNames() async {
  final prefs = MockSharedPreferences.getInstance();
  await prefs.remove('userNames');
  setState(() {
    _userNames = [];
  });
  _showSnackBar("Tüm kullanıcı adları silindi!");
}

// Belirli bir kullanıcı adını sil
_removeName(int index) async {
  setState(() {
    _userNames.removeAt(index);
  });

  final prefs = MockSharedPreferences.getInstance();
  await prefs.setString('userNames', jsonEncode(_userNames));
  _showSnackBar("Kullanıcı adı silindi!");
}
```

Bu metotlar:

1. `_clearNames()` tüm kayıtlı kullanıcı adlarını siler
2. `_removeName(int index)` belirli bir indeksteki kullanıcı adını siler
3. Her iki işlem sonrasında da SharedPreferences güncellenip kullanıcıya bildirim gösterilir

Kullanıcı Arayüzü

Uygulamanın kullanıcı arayüzü, basit ancak işlevsel bir tasarıma sahiptir:

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('SharedPreferences Örneği'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          Text(
            'Dart Temelleri: SharedPreferences',
            style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
            textAlign: TextAlign.center,
          ),
          const SizedBox(height: 24),
          TextField(
            controller: _nameController,
            decoration: const InputDecoration(
              labelText: 'Kullanıcı Adı',
              border: OutlineInputBorder(),
            ),
          ),
          const SizedBox(height: 16),
          ElevatedButton(
            onPressed: _saveName,
            child: const Text('Kaydet'),
          ),
          // Diğer UI elemanları...
        ],
      ),
    ),
  );
}
```

Arayüz şu bileşenlerden oluşur:

1. Başlık ve açıklama metni
2. Kullanıcı adı girişi için TextField
3. Kaydetme butonu
4. Kaydedilen kullanıcı adlarını listeleyen bir ListView
5. Silme işlemleri için butonlar

6. Kullanıcı geri bildirimleri için SnackBar mesajları

Uygulamanın Çalışması

Uygulama çalıştırıldığında şu adımlar gerçekleşir:

1. Önce `initState()` içinde `_loadSavedNames()` metodu çağrılarak daha önce kaydedilmiş kullanıcı adları yüklenir
2. Kullanıcı, metin alanına bir ad girebilir ve "Kaydet" butonuna tıklayabilir
3. Başarılı kaydetme işlemi sonrasında girilen ad listeye eklenir ve SharedPreferences'a kaydedilir
4. Kullanıcı, bireysel olarak her adın yanındaki çöp kutusu simgesine tıklayarak o adı silebilir
5. "Tümünü Sil" butonu ile tüm liste temizlenebilir
6. Her işlem sonrası kullanıcıya SnackBar ile bildirim gösterilir

Sonuç

Bu proje, Flutter ve Dart kullanarak mobil uygulamalarda yerel veri depolama işlemlerinin nasıl gerçekleştirilebileceğini göstermektedir. SharedPreferences API'sinin kullanımını öğrenmek, mobil uygulama geliştirme sürecinde önemli bir adımdır.

Projede gösterilen teknikler, kullanıcı tercihlerini saklamak, oturum bilgilerini yönetmek veya çevrimdışı verilerle çalışmak gibi birçok gerçek dünya uygulamasında kullanılabilir.

Gelecek geliştirmeler için düşünülebilecek bazı eklemeler:

- Kullanıcı adlarının yanı sıra diğer bilgilerin (e-posta, yaş vb.) saklanması
- Verilerin şifrelenmesi
- Bulut senkronizasyonu
- Daha gelişmiş veri doğrulama kontrolleri

Bu proje, daha karmaşık veri yönetimi senaryoları için temel bir başlangıç noktası olarak kullanılabilir.