/\* 1) İlk başta Unity içinde tutmak istemediğiniz dosyaları prefab halina getirmeliyiz. Prefab yaptığımız dosyanın üzerine gelip Inspector penceresinin en alt kısmında AssetBundle olarak bulunan kısımda tıklayarak prefab objenin assetbundle ismini vermeliyiz.\*/

/\* 2) Prefab dosyalarımıza verdiğimiz assetbunlde isimlerini bir dosya haline getirip internete atmamız gerekiyor. Bunun için ;

Unity assets klasörü içinde Editor adında bir dosya oluşturup bunun içinde CreateAssetBundle.cs adında bir script açıyoruz. bu scriptin içine

AssetBundle'ın adını Unity içinde Assets'in altında StreamingAssets Adında bir dosya oluşturacak ve burda prefablara verdiğiniz assetBundle isimlerini görebileceksiniz.\*/

|  |
| --- |
| **using** **System.Collections**;  **using** **System.Collections.Generic**;  **using** **UnityEngine**;  **using** **System.IO**;  **using** **UnityEditor**;  **public** **class** **CreateAssetBundles**  {  [MenuItem("Assets/Build AssetBundles")]  **static** **void** **BuildAllAssetBundles**()  {  **string** assetBundleDirectory = "Assets/StreamingAssets";  **if** (!Directory.Exists(Application.streamingAssetsPath))  {  Directory.CreateDirectory(assetBundleDirectory);  }  /\*burda bir hat oluşturuyor BuildAssetsBundlestan ve oluşabilecek bütün hedef olan platformlarda çekebileceksiniz. Yani android,ios vb gibi \*/  BuildPipeline.BuildAssetBundles(assetBundleDirectory, BuildAssetBundleOptions.None, EditorUserBuildSettings.activeBuildTarget);  }  } |

/\*3) AssetBundles Klasörünü oluşturabilmek için Unity üzerinden Assets sekmesinde en altta build AssetsBundles adını göreceksiniz ona tıkladığınızda 20-60 saniye aralığında size bir dosya oluşturacak ve bu dosya Project/Assets/StreamingAssets adı altında oluşmuş olacaktır. \*/

/\*4) Oluşturduğumuz "StreamingAssets" adındaki assetBundle unity dosyalarının içinden çıkartıp masaüstümüze alıyoruz. Aldıktan sonra bir online sunucu oluşturup bu sunucu içine bu dosyaları atmamız gerekiyor. Dosyaların VerdiğimizAssetBundleName.manifest olan dosyamız bizim version numarasına ulaşacağımız dosyadır. Yalnızca Verdiğimiz AssetBundleName dosyamız ise bizim internet üzerinden çekeceğimiz prefab dosyamızdır.\*/

/\* 5) Attığımız dosyaları nasıl çekeriz?

Atılan dosyaları yeni bir API adlı script oluşturarak bu scriptin içine   
altta verilen kodları ekliyoruz yanda açıklamaları yazmaktadır\*/

|  |
| --- |
| **using** **System.Collections**;  **using** **UnityEngine**;  **using** **UnityEngine.Events**;  **using** **UnityEngine.Networking**;  **using** **System.Collections.Generic**;  **using** **System.IO**;  **using** **System**;  **public** **class** **API** : MonoBehaviour  {  **const** **string** BundleFolder = "Sanal sunucu url adresi" /\*Örnek: "http://www.ndgstudio.com.tr/berkcan/"\*/  **public** List<**string**> LoadingObjectValue = **new** List<**string**>();  /\*yüklenecek objelerimizin listesi\*/  **public** List<Hash128> listOfCachedVersions = **new** List<Hash128>();  /\*yüklenecek objelerin version numaraları \*/  **public** AssetBundle bundle; /\*çekilecek assetbundle ismi\*/  **string** bundleURL; /\* dosyalarımız içindeki verdiğimiz AssetBundleName dosyamızın ismini aratmak için bir string tanımlıyoruz\*/  **public** UnityWebRequest www; /\* Web request atmamız için gerekli olan www adında tanımlandı.\*/  **public** **void** **GetBundleObject**(UnityAction<GameObject> callback, Transform bundleParent)  {  StartCoroutine(GetDisplayBundleRoutine(callback, bundleParent));  }  IEnumerator **GetDisplayBundleRoutine**(UnityAction<GameObject> callback, Transform bundleParent)  {  **for** (**int** i = **0**; i < listOfCachedVersions.Count; i++)  {  /\*verdiğimiz objenin adını üstteki bundle folder yani dosyanın konumu  verdiğimiz assetbundleName'i giriyoruz\*/  (örnek1: **1**-Android örnek2: Berkcan-Android)  bundleURL = BundleFolder + LoadingObjectValue[i];  /\* burda platformu tanımlayıp sonuna android veya ios sa eklemesi için verilen bir yer dosyamızın ismi 1-Android verdiysek üstte bundleURL = BundleFolder + "1-" olarak verirsek alttaki ile "1-Android" olacaktır. Bunun için oluşturduğumuz prefab dosyasına isim verirken assetbundle kısmında isminin sonuna android veya IOS eklememiz gerekli eğer android veya ios platform yazmasını istemiyorsak alttaki satırı commendleyip devam edebilirsiniz.\*/  #if UNITY\_ANDROID  bundleURL += "Android";  #else  bundleURL += "IOS";  #endif  /\* burada bundleURL adındaki assetbundle dosyamızı www ye eşitleyip bir istek gönderiyoruz. \*/  www = UnityWebRequestAssetBundle.GetAssetBundle(bundleURL);  **yield** **return** www.SendWebRequest();  /\*www adında işleyici varlık paketini indirip bundle'a eşitliyoruz\*/  bundle = DownloadHandlerAssetBundle.GetContent(www);  Debug.Log("Requesting bundle at " +bundleURL);  }  Debug.Log("BundleUrl : " + bundleURL);  /\* eğer internete bağlı değilse olacaklar veya dosyaya ulaşamazsa \*/  **if** (www.isNetworkError || www.isHttpError)  {  Debug.Log(www.error);  }  **else**  {  /\* eğer dosyaya ulaşırsa olacaklar\*/  /\* alttaki ifte sizden dosya uzantısının adını ve version numarasını istiyor bunun için BundleURL ve benim listemdeki version numaralarını listeden giriyorum. Version numarası için AssetBundleName.manifest dosyasının içindeki AssetFileHash: Altındaki Hash: kodunu kopyalayıp koyuyoruz.\*/  **if** (Caching.IsVersionCached(bundleURL, listOfCachedVersions[i]))  {  **string** rootAssetPath = bundle.GetAllAssetNames()[**0**];  GameObject arObject = Instantiate(bundle.LoadAsset(rootAssetPath) **as** GameObject, bundleParent); callback(arObject);  }  /\* Eğer bu version numarası cache de yoksa buraya giriyor \*/  **else** **if** (!Caching.IsVersionCached(bundleURL, listOfCachedVersions[i]))  {  /\* bundle unload yapıyoruz.\*/  **if** (bundle != **null**)  {  bundle.Unload(**false**);  }  **while** (!Caching.ready)  **yield** **return** **null**;  /\* burada ise cacheden veya dosyayı direkt olduğu konumdan indirip oyunumuza instantiate ediyoruz. \*/  **using** (**var** wwww = WWW.LoadFromCacheOrDownload(bundleURL, listOfCachedVersions[i]))  {  **yield** **return** wwww;  **if** (!**string**.IsNullOrEmpty(wwww.error))  {  Debug.Log(wwww.error);  **yield** **return** **null**;  }  bundle = wwww.assetBundle;  **string** asset = bundle.GetAllAssetNames()[**0**];  GameObject game = Instantiate(bundle.LoadAsset(asset) **as** GameObject, **new** Vector3(**0**, **0**, **0**), Quaternion.identity);  callback(game);  }  }  }  }  }  } |

/\* UNUTMAYIN LİSTE YÖNTEMİ İLE YAPTIĞIMIZ İÇİN VERSİON NUMARASI VE BUNDLE İSMİNİ GİRMENİZ GEREKİYOR.  
ContentController adında bir script oluşturup aşağıdaki kod satırlarını girmeniz gerekiyor. Bu scrtip içerikleri denetleyip yüklenen objenin ismini yazdırıyor. Aşağıdaki LoadContent() fonksiyonunu bir buton veya startta çağırarak scripti başlatabilirsiniz. \*/de

|  |
| --- |
| **using** **UnityEngine**;  **using** **System.Collections**;  **using** **System.Collections.Generic**;  **using** **System.IO**;  **using** **UnityEngine.Networking**;  **using** **System**;  **public** **class** **ContentController** : MonoBehaviour {  **public** API api;  **public** **void** **LoadContent**()  {  DestroyAllChildren();  api.GetBundleObject(OnContentLoaded, transform);  }  **void** **OnContentLoaded**(GameObject content) {  //do something cool here  Debug.Log("Loaded: " + content.name);  }  **void** **DestroyAllChildren**() {  **foreach** (Transform child **in** transform) {  Destroy(child.gameObject);  }  }  } |