



RUST **Struct** **HashMap** **Vektör** **birlikte kullanımı**



HashMap<K,V>

HashMap veri yapısında sadece iki veri yapısı kullanılabilmektedir.

Struct yapısını kullanarak farklı ve birden fazla veri yapılarını HashMap içerisinde tutabiliriz.

- Struct'ın key-anahtar olarak kullanılması
- Struct'ın value-değer, yani veri olarak kullanılması



Struct Kullanarak HashMap Oluşturma:

```
#![allow(unused)]
fn main() {
    use std::collections::HashMap;

    #[derive(Hash, Eq, PartialEq, Debug)]
    struct Viking {
        name: String,
        country: String,
    }

    impl Viking {
        /// Creates a new Viking.
        fn new(name: &str, country: &str) -> Viking {
            Viking { name: name.to_string(), country: country.to_string() }
        }
    }
}
```



```
// Use a HashMap to store the vikings' health points.  
let vikings = HashMap::from([  
    (Viking::new("Einar", "Norway"), 25),  
    (Viking::new("Olaf", "Denmark"), 24),  
    (Viking::new("Harald", "Iceland"), 12),  
]);  
  
// Use derived implementation to print the status of the vikings.  
for (viking, health) in &vikings {  
    println!("{viking:?} has {health} hp");  
}  
}
```



KEY

VALUE

{ kullanıcı_bilgileri : kullanıcı_no } şeklinde bir hashmap oluşturmaya çalışalım.

Kullanıcı_bilgileri → username, mail, active

```
struct User {  
    active: bool,  
    username: String,  
    mail: String,  
}
```



```
let users = HashMap::from([(
    User {
        active: false,
        username: "c11".to_string(),
        mail: "c11@rust".to_string(),
    },
    1,
)]);
```

HashMap tanımlayıp User yapısını key-anahtar olarak verdiğimizde hata oluşur.

```
#[derive(Hash, Eq, PartialEq, Debug)]
```

Hash → Struct üzerinde hash fonksiyonun uygulanmasını sağlar

Eq, PatialEq → Karşılaştırma işlemlerinin yapılmasını sağlar. Key değer tek olmak zorundadır ve ekleme işlemi yapılırken karşılaştırma yapılacaktır.



Verileri daha kısa bir şekilde yazabilmek için metod kullanabiliriz.

```
impl User {  
    fn new(username: &str, mail: &str) -> User {  
        User {  
            active: false,  
            username: username.to_string(),  
            mail: mail.to_string(),  
        }  
    }  
}
```

```
( User::new("aks", "aks@gmail.com"), 3),
```



Struct yapısını veri olarak kullandığımız zaman böyle bir sorun ile karşılaşmayız.

Çünkü struct üzerinde, key-anahtar için yapılan işlemler yapılmayacaktır.



```
#[derive(Hash, Eq, PartialEq, Debug)]
struct User {
    active: bool,
    username: String,
    mail: String,
}

impl User {
    fn new(username: &str, mail: &str) -> User {
        User {
            active: false,
            username: username.to_string(),
            mail: mail.to_string(),
        }
    }
}
```



```
use std::collections::HashMap;
fn main() {
    let users = HashMap::from([
        (
            1,
            User {
                active: false,
                username: "c11".to_string(),
                mail: "c11@rust.com".to_string(),
            },
        ),
        (
            2,
            User {
                active: false,
                username: "rust".to_string(),
                mail: "rust@rust.com".to_string(),
            },
        ),
        (3, User::new("aks", "aks@gmail.com")),
    ]);

    println!("{:#?}", users);
}
```

HashMap Kullanarak Struct Oluşturma:



```
struct Sınıf {  
    ogrenciler: HashMap<String, String>,  
    seviye: String,  
}
```

```
let a9 = Sınıf {  
    ogrenciler: HashMap::from([  
        ("132".to_string(), "ahmet".to_string()),  
        ("144".to_string(), "fatma".to_string()),  
    ]),  
    seviye: "9a".to_string(),  
};
```

```
a9.ogrenciler.insert("155".to_string(), "ayşe".to_string());
```



<https://doc.rust-lang.org/stable/std/collections/struct.HashMap.html>



Vectorlerde Struct Kullanımı :

```
let mut users1: Vec<User> = Vec::new();  
users1.push(User {  
    active: false,  
    username: "ddd".to_string(),  
    mail: "ddd".to_string(),  
});
```



Struct içinde Vector Kullanımı :

```
struct Sınıf2 {  
    ogrenciler: Vec<String>,  
    seviye: u32,  
}
```

```
let mut vec_9a = Sınıf2 {  
    ogrenciler: vec!["ahmet".to_string(), "fatma".to_string()],  
    seviye: 1,  
};  
  
vec_9a.ogrenciler.push("belkıs".to_string());
```




Celal AKSU

Bilişim Teknolojileri Öğretmeni

celalaksu@gmail.com

<https://www.linkedin.com/in/cllaksu/>

<https://twitter.com/ksacil>

<https://www.youtube.com/@eemcs>