# Tapestry5HowToIocAndHibernate

source: mini_app_hibernate.zip

This article continues from [Tapestry5HowToIocOnly] and adds tapestry-hibernate to the app.

Tapestry hibernate is currently dependant on tapestry-core (which will hopefully be fixed later).

We will add a dependancy to tapestry-core, but will not load whole TapestryModule.

Only some parts of the code will be provided in the page, and the rest as attachment.

For tapestry-hibernate to work without tapestry-core you need some extra setup in AppModule.

definitely make sure to call `registry.cleanupThread();` at the end of the main method or your data will not be saved (tapestry-hibernate performs session.commit() when that event occurs)

```
public class MiniAppModule {

    public static void bind(ServiceBinder binder){
        binder.bind(Hello.class);
        //tapestry-hibernate needs this dependency
        binder.bind(ClassNameLocator.class, ClassNameLocatorImpl.class);
    }

    public static void contributeSymbolSource(OrderedConfiguration<SymbolProvider> conf){
        //tapestry-hibernate fails without tapestry.app-name symbol defined
        conf.add("AppPackage", new SymbolProvider(){
           public String valueForSymbol(String symbolName){
               if(symbolName.equalsIgnoreCase(InternalConstants.TAPESTRY_APP_PACKAGE_PARAM))
                   return "tapestry.mini";
               return null;
            }
        },"");
    }

    public void contributeHibernateEntityPackageManager(Configuration<String> configuration)
    {
//      extra packages...
//      tapestry-hibernate will add entities package automaticaly for TAPESTRY_APP_PACKAGE_PARAM+".entities"
//      configuration.add("tapestry.mini.entities");
    }
}
```

Hello class is bit different for this demo

```
public class Hello {

    private final UserDao _userDao;

    public Hello(UserDao userDao){
        _userDao = userDao;
    }

    public void sayHello(){
        List<User> list = _userDao.findAll();
        if(list.size() == 0)
            System.out.print("No users in database");
        else
            System.out.print("Hello "+list.get(0).getName());
    }
}
```

User entity is just something simple

```
@Entity
public class User {

    Long id;
    String name;
    String username;

    public User(){}
    public User(String name, String username) {
        super();
        this.name = name;
        this.username = username;
    }

    @Id @GeneratedValue
    public Long getId() {
        return id;
    }
//... other getters/setters
}
```

User dao just extends Generic Data Access Objects from http://www.hibernate.org

```
public class UserDAO extends GenericHibernateDAO<User, Long> {
    //if you forget this constructor Session will not be injected
    public UserDAO(Session session) {super(session);}

    //example for search by criteria
    public List<User> findByName(String name){
        //findByCriteria is hibernate specific so you can not call _userDao.findByCriteria from
        //outside this class, and you should not expose it either
        return this.findByCriteria(Restrictions.eq("name", name));
    }
}
```

run the app twice, first time it will create an user in the database, second time it will say Hello John.