# TASKZERO – **STEP 5**

This lab starts from the source code of Step2 (which was created following suggestions found in lab4.pdf). As first thing, make a new **copy of the project in Step2** and rename it appropriately. The purpose of the lab is adding a view that shows the history of a given task within the application.

- To start off, set the version number of the demo. Open **TaskZeroSettings.cs** and change the **Version** constant to 3.
- Open web.config and make sure the authentication cookie is named **TASKZERO3**.

If you compile the project, you should see the same app as in Step 2 but referring to step 3.

- Let's first edit the user interface. Open **pv_PendingTasks.cshtml** under **Views/Dashboard** and add a third button to each row in the task list. The third button will be used to trigged the view that presents the history of the task.

```
<a role="button" class="btn btn-primary"
        href="@Url.Action("history", "task", new {id=task.TaskId})">
    <i class="fa fa-fw fa-history"></i>
</a>
```

- Open **global.asax.cs** and add the following line

```
        public static IEventStore EventStore { get; private set; }
```

  in the same file, in the code of **Application_Start** make sure you have the following code:

```
        Bus = container.Resolve<IBus>();
        AggregateRepository = container.Resolve<IRepository>();
        EventStore = container.Resolve<IEventStore>();
```

- In the **CommandStack** project, within the **Model** folder create a **TaskHistory.cs** file.

```
namespace Mfx3.CommandStack.Model
{
    public class TaskHistory
    {
        public TaskHistory(Guid taskId, DateTime when,
                           IEnumerable<TaskTransition> transitions)
        {
            TaskId = taskId;
            When = when;
            Events = transitions;
        }

        public Guid TaskId { get; set; }
        public DateTime When { get; set; }
        public IEnumerable<TaskTransition> Events { get; }
    }
}
```

- In the same folder also creates a **TaskTransition.cs** file.

```
namespace TaskZero.CommandStack.Model
{
    public class TaskTransition
    {
        public TaskTransition(string action, DateTime when, Task temp)
        {
            Action = action;
            When = when;
            CurrentTask = temp;
        }

        public string Action { get; set; }
        public DateTime When { get; set; }
        public Task CurrentTask { get; set; }
    }
}
```

- In the folder **Models/Task**, create a new file **TaskHistoryViewModel.cs** as below:

```
using TaskZero.CommandStack.Model;

namespace TaskZero.Models.Task
{
    public class TaskHistoryViewModel : ViewModelBase
    {
        public TaskHistory History { get; set; }
    }
}
```

- Open **TaskController.cs** and add a **History** method to handle the click on the History button.

```
#region HISTORY TASK
[HttpGet]
public ActionResult History(string id)
{
    Guid guid;
    var outcome = Guid.TryParse(id, out guid);
    if (!outcome)
        throw new InvalidGuidException("Could not find specified task");

    var history = new HistoryService(MfxApplication.EventStore,
                                     MfxApplication.AggregateRepository);
    var model = new TaskHistoryViewModel
    {
        History = history.GetTaskHistory(guid, DateTime.Now)
    };
    return View(model);
}
#endregion
```

- Create a folder **Services** in the CommandStack project and add a couple of new files called DomainService.cs and **HistoryService.cs**. The former is the base class of the latter which will contain the logic to retrieve historical data.

```
public class DomainService
{
        public DomainService(IEventStore eventStore, IRepository repository)
        {
            EventStore = eventStore;
            Repository = repository;
        }

        public IEventStore EventStore { get; private set; }
        public IRepository Repository { get; private set; }
}
```

The **HistoryService.cs** file looks like below:

- In the same folder, also add a **DomainEventExtensions.cs** class

```
public static class DomainEventExtensions
{
        public static string ShortName(this DomainEvent theEvent)
        {
            var type = theEvent.GetType().ToString().ToLower();
            if (type.Contains("created"))
                return "CREATED";
            if (type.Contains("completed"))
                return "COMPLETED";
            if (type.Contains("deleted"))
                return "DELETED";
            if (type.Contains("updated"))
                return "UPDATED";
            return "";
        }
}
```

- Add a **TaskExtensions.cs** helper class to the **Common/Extensions** folder of the server project.

```
public static class TaskExtensions
{
    public static string ToColor(this PendingTask pendingTask,
                                      Priority priority)
    {
        switch (priority)
        {
            case Priority.Urgent:
                return "#f00";
            case Priority.High:
                return "#f80";
            case Priority.Normal:
                return "#0c0";
            case Priority.Low:
                return "#0f8";
            default:
                return "transparent";
        }
    }
}
```

- As final step, let's add the view for historical data. Add a **history.cshtml** file to the **Views/Task** folder.

```
@using Expoware.Youbiquitous.Extensions
@using TaskZero.Server.Common.Extensions
@model TaskZero.Server.Models.Task.TaskHistoryViewModel


<div class="col-xs-12 col-lg-10 col-lg-offset-1">
    <h2>
        <a href="@Url.Action("index", "dashboard")">
               <i class="fa fa-history"></i></a>
           HISTORY
        <small class="text-muted hidden-xs">@Model.History.TaskId</small>
    </h2>

    <table class="table table-condensed">
        <thead>
            <tr class="bold" style="font-size: 120%">
                <td>ACTION</td>
                <td style="width: 10px"></td>
                <td>TASK</td>
                <td>STATUS</td>
                <td>DUE DATE</td>
                <td>&nbsp;</td>
            </tr>
        </thead>
        <tbody>
            @foreach (var ev in Model.History.Events)
            {
                <tr>
                    @if (ev.Action == "DELETED" || ev.Action == "COMPLETED")
                    {
                        <td>
                            <b class="text-primary">@ev.Action</b><br />
                            <small class="text-
muted">@ev.When.ToString("d MMM yyyy HH:mm") UTC</small>
                        </td>
                        <td colspan="4"></td>
                    }
                    else
                    {
                        <td>
                            <b class="text-primary">@ev.Action</b><br />
                            <small class="text-
muted">@ev.When.ToString("d MMM yyyy HH:mm") UTC</small>
                        </td>

                        <td style="background:
                      @ev.CurrentTask.ToColor(ev.CurrentTask.Priority)"></td>
```

```
                        <td>
                            @ev.CurrentTask.Title<br />
                            <small class="text-
muted">@ev.CurrentTask.Description</small>
                        </td>
                        <td>
                            @ev.CurrentTask.Status
                        </td>

                        <td>
                            @ev.CurrentTask.DueDate.ToStringOrEmpty("d MMM yyyy")
                        </td>
                    }
                </tr>
            }
        </tbody>
    </table>
</div>
```