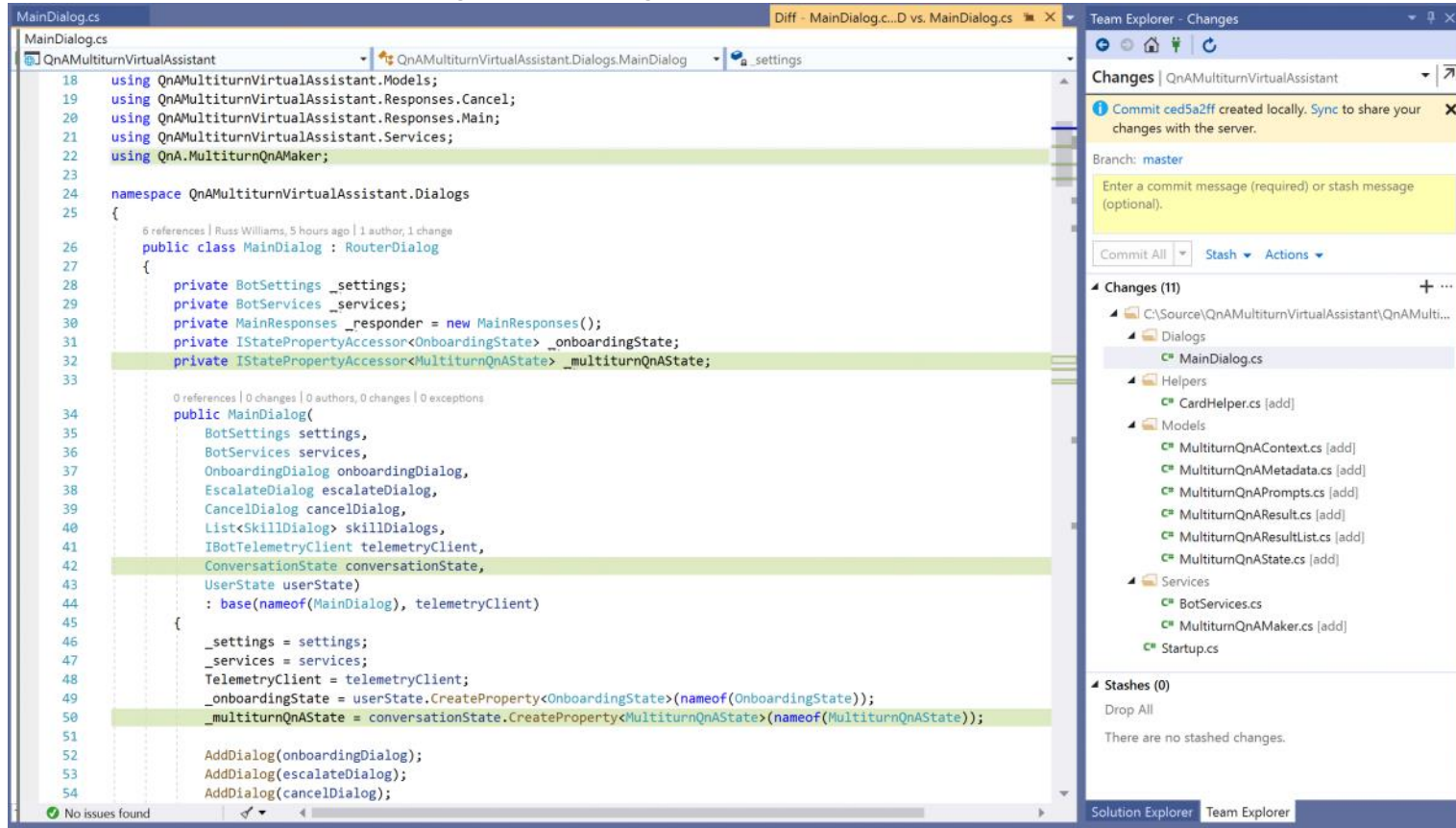


Changes to MainDialog.cs

Thursday, June 13, 2019 1:01 AM

There are two sets of changes you need to make to the MainDialog.cs file.

Diff screen shot to show first set of changes to MainDialog.cs



Snippet for easy cut & paste (copy bolded lines)

```
using QnA.MultiturnQnAMaker;

namespace QnAMultiturnVirtualAssistant.Dialogs
{
    public class MainDialog : RouterDialog
    {
        private BotSettings _settings;
        private BotServices _services;
        private MainResponses _responder = new MainResponses();
        private IStatePropertyAccessor<OnboardingState> _onboardingState;
        private IStatePropertyAccessor<MultiturnQnAState> _multiturnQnAState;

        public MainDialog(
            BotSettings settings,
            BotServices services,
            OnboardingDialog onboardingDialog,
            EscalateDialog escalateDialog,
            CancelDialog cancelDialog,
            List<SkillDialog> skillDialogs,
            IBotTelemetryClient telemetryClient,
            ConversationState conversationState,
            UserState userState)
            : base(nameof(MainDialog), telemetryClient)
        {
            _settings = settings;
            _services = services;
            TelemetryClient = telemetryClient;
            _onboardingState = userState.CreateProperty<OnboardingState>(nameof(OnboardingState));
            _multiturnQnAState = conversationState.CreateProperty<MultiturnQnAState>(nameof(MultiturnQnAState));

            AddDialog(onboardingDialog);
            AddDialog(escalateDialog);
            AddDialog(cancelDialog);
        }
    }
}
```

Diff screen shot to show second set of changes to MainDialog.cs

```
Diff - MainDialog.c...D vs. MainDialog.cs
MainDialog.cs
QnAMultiturnVirtualAssistant
QnAMultiturnVirtualAssistant.Dialogs.MainDialog
RouteAsync(DialogContext dc, CancellationToken cancellationToken = def

132         await _responder.ReplyWith(dc.Context, MainResponses.ResponseIds.Confused);
133         break;
134     }
135 }
136 }
137 }
138 else if (intent == DispatchLuis.Intent.q_faq)
139 {
140     cognitiveModels.QnAServices.TryGetValue("faq", out var qnaService);
141
142     if (qnaService == null)
143     {
144         throw new Exception("The specified QnA Maker Service could not be found in your Bot Services configuration.");
145     }
146     else
147     {
148         MultiturnQnAState oldState = await _multiturnQnAState.GetAsync(dc.Context);
149
150         // Comment out original call to GetAnswersAsync
151         //var answers = await qnaService.GetAnswersAsync(dc.Context, null, null);
152
153         (var newState, var answers) = await ((MultiturnQnAMaker)qnaService).GetAnswersAsync(oldState, dc.Context, null, null);
154
155         if (answers != null && answers.Count() > 0)
156         {
157             await dc.Context.SendActivitiesAsync(answers.ToArray());
158         }
159         else
160         {
161             await _responder.ReplyWith(dc.Context, MainResponses.ResponseIds.Confused);
162         }
163
164         await _multiturnQnAState.SetAsync(dc.Context, newState, cancellationToken);
165     }
166 }
167 else if (intent == DispatchLuis.Intent.q_chitchat)
168 {
169     cognitiveModels.QnAServices.TryGetValue("chitchat", out var qnaService);
170 }
```

Snippet for easy cut & paste (copy bolded lines)

```
else if (intent == DispatchLuis.Intent.q_faq)
{
    cognitiveModels.QnAServices.TryGetValue("faq", out var qnaService);

    if (qnaService == null)
    {
        throw new Exception("The specified QnA Maker Service could not be found in your Bot Services configuration.");
    }
    else
    {
        MultiturnQnAState oldState = await _multiturnQnAState.GetAsync(dc.Context);

        // Comment out original call to GetAnswersAsync
        //var answers = await qnaService.GetAnswersAsync(dc.Context, null, null);

        (var newState, var answers) = await ((MultiturnQnAMaker)qnaService).GetAnswersAsync(oldState, dc.Context, null, null);

        if (answers != null && answers.Count() > 0)
        {
            await dc.Context.SendActivitiesAsync(answers.ToArray());
        }
        else
        {
            await _responder.ReplyWith(dc.Context, MainResponses.ResponseIds.Confused);
        }

        await _multiturnQnAState.SetAsync(dc.Context, newState, cancellationToken);
    }
}
```