## Pluralsight Course Proposal Template

*Updated June 4, 2018*

*Proposing a developer course? Here’s an* [*example*](https://docs.google.com/a/pluralsight.com/document/d/1QhIQ79Tvip9iSJfk3cwtXCVKprpsitpkf_Lxd6tpjdw)*.*

*Proposing an IT admin course? Here’s your* [*example*](https://docs.google.com/a/pluralsight.com/document/d/1v6mBtHNSoIs1Dme13rSW78IjBvGfAvKlu6pJPvAhkfo)*.*

*Proposing a Creative | Design | Engineering course? Here’s your* [*example*](https://docs.google.com/a/pluralsight.com/document/d/17gZb5iWdW2frtuZjypOLjcyXoSe3Re27lnZ0YaQ8oyc/edit?usp=sharing)*.*

## *Visit*[*https://authors.pluralsight.com/proposing-course*](https://authors.pluralsight.com/proposing-course/) *for more details and tips.*

|  |  |
| --- | --- |
| **TITLE**  Max 65 characters  [Course naming conventions.](https://authors.pluralsight.com/proposing-course/course-types-dev/#course-types) | C# Design Patterns: Null Object |
| **AUTHOR** | David Starr |
| **LEVEL**  Reflects the number of courses someone must watch, or pre-req knowledge they must have, before watching this one. [Beginner, Intermediate, or Advanced](https://authors.pluralsight.com/proposing-course/course-types-dev/#audience-levels) | 200 |
| **DURATION**  Estimate in hours | 0.5 |
| **AUDIENCE/INDUSTRY TAG**  Usually only 1; indicates primary audience/industry. Choose from:  software-development  data-professional  It-ops  security-professional  business-professional  architecture-construction  manufacturing-design  creative-professional | Use whichever of these are being applied to patterns.  software-development  architecture-construction |
| **AUDIENCE PROFILE**  Who should watch this course? What kinds of job roles do they hold in their organization? | Software developers wanting to improve their software design skills. |
| **ABSTRACT**  Should “sell” the course. Treat this as if it will appear on the website. Refer to [this page in the author kit](https://authors.pluralsight.com/course-deliverables/long-description/) for guidance on writing concise & informative "long" course descriptions. | Null objects can be a real pain to work with, but what if you could greatly reduce the null checks in your code? That would be great, right? Meet the Null Object pattern.  Software design patterns are repeatable known solutions to regularly occurring code design problems. They also provide a common language for use to use in speaking about our code. |
| **PREREQUISITES**  Describe skills and knowledge (not specific courses) that someone should have before watching this course. | Static language experience like Java or C# and .NET Core |
| **RELATIONSHIP TO EXISTING COURSES**  Provide links to related courses and how this one fits in, or describe the content gaps that this course fills | Sit alongside other courses used to create a single search for all design patterns. |
| **DESCRIPTION OF SAMPLE PROJECT / SCENARIO**  Describe the overarching scenario(s) that your examples and demonstrations will come from. Be specific. | The user will see code that is poorly implemented and would benefit from using this pattern. “Do you ever find yourself in this situation?” A pattern can help here, cleaning this code up and making things more testable. |
| **PLATFORM/TOOL VERSIONS**  Indicate which version(s) of platform/software/hardware you plan to **demonstrate or use** in your course, including whether or not any will still be in pre-release when you record. Note that in some cases it may be appropriate to specify multiple versions or even a range.  Additionally, please include your thoughts on how compatible the techniques or code you plan to demonstrate are with older versions of the relevant products/platforms. | |  |  |  | | --- | --- | --- | | **Technology** | **Version(s)** | **Pre-release? (Y/N)** | | C# and .NET Core | .NET Core 3 | N |   Due to the generically applicable solution in this course it is appropriate for multiple static languages and versions. including C# and Java, current and past version.  All demo will be in C# and .NET Core 3 |

|  |  |  |
| --- | --- | --- |
| **Module Outline** | | |
| **1** | **Course Overview**  *Do not edit or remove this module from the course outline.*  You will need to narrate a very brief informational [course overview](https://authors.pluralsight.com/course-deliverables/course-trailer/) toward the end of course production in preparation for publication.  Course overviews are the first module of your course and are accessible outside the Pluralsight paywall without a subscription. Overviews are used to market courses and provide an easy way for learners to quickly determine whether a course is right for them.  It’s a good idea to read over the course overview script now. That way you know what will and will not be covered, and can avoid accidentally repeating the same information elsewhere in the course.  No action is required during the proposal process. Your Production Editor will coordinate this with you when the time comes, and your Author Success Manager can answer any questions you may have at this time. | **60 - 90 seconds** |
| **1** | **Introduction**  Tell the story of what null is any why we have it in OOP languages including the inventor, Tony Hoare. Talk about the pain it causes in out codebase and the money it costs in industry.  **Demo – Show the pain**  Show a constructor with 2 null checks at the top for 2 parameters it is receiving. Take away a null check and recreate it using the built-in capability of the IDE. This shows how the IDE even has built in functionality for creating null check branches given what a pervasive pattern it is to perform null checks.  Discuss it, and state there must be a better way. In many cases, there is a better way and it’s using the Null Object pattern.  **Slide on complexity**  Explain cyclomatic complexity and why higher scores are bad. Explain the impact of the code we saw on cyclomatic complexity. We can often reduce this score, and thus the complexity of our code, using the Null Object pattern.  **Introduce the Null Object pattern**  Introduce the Null Object pattern conceptually. First with a UML model of the pattern, then a less formal view of the pattern without UML. Show its shape and structure using a concrete type and the inheritance model of Null Objects.  **Demo - Applying the Null Object Pattern**  Go back to the IDE and walk through replacing one of the 2 null checks with a NullObject. Note our cyclomatic complexity just reduced by one.  Now the object creating our class of interest will pass in a NullObject. Note this doesn’t mean it’s really null. It means it has default values. If one were to rename this pattern, it could be know as the Default pattern because it provides default values instead of nulls.  Also note that  **Slides**  Show the visual concept of having solved the problem with the pattern. Note the pattern is not appropriate for all situations. Note it isn’t a panacea and there are cases for which it should not be used. For example, not to be returned as an element in a set, like a list or array.  **Wrapping Up**  Reminder of the pattern’s intended usage  Recall the reduction in null checks, reducing complexity  Using the NO pattern, we write cleaner, simpler code that is easier to read and maintain. | **~20m** |