Andrew Downes
andownes@gmail.com

Expose Additional QuantLib Template Classes to QuantLibXL

This document demonstrates how to expose a template QuantLib class to excel. It is assumed that you have already compiled the full build of QuantLibXL.

Here we expose the QuantLib class PiecewiseZeroInflationCurve. The method used is based on the PiecewiseYieldCurve class which has already been exposed to Excel. This document outlines the necessary steps and gives most of the required code base, for a more detailed explanation see the corresponding files for the PiecewiseYieldCurve. See also http://quantlib.org/quantlibaddin/extend_tutorial.html for a tutorial on exposing regular QuantLib classes to QuantLibXL.

The PiecewiseZeroInflationCurve class can have up to three template arguments. However, for this tutorial we only consider the first template argument, we use the default template type for the second and third arguments.

Some additional classes such as the InflationRateHelper class and ZeroInflationTermStructure class are required to construct a PiecewiseZeroInflationCurve, as such a brief outline of the files and class names used to expose these to Excel is given. Steps 3-6 are the main steps relating to wrapping a template class.

1. Expose the InflationRateHelper class

This step is included for completeness, not because it has any relevance to template classes. If we wish to use our PiecewiseZeroInflationCurve in Excel we will need to be able to create InflationRateHelper objects. A tutorial for exposing classes such as this one can be found at http://quantlib.org/quantlibaddin/extend_tutorial.html and we will not repeat everything written there in this tutorial. We do provide an outline of the files and code to ensure later code is consistent.

The code for these classes is written in the following two new files:

```
QuantLibAddin\qlo\inflationratehelper.hpp
QuantLibAddin\qlo\inflationratehelper.cpp
```

An outline of the code in the header file is given below:

```
#include <oh/libraryobject.hpp>

#include <ql/termstructures/inflation/piecewisezeroinflationcurve.hpp>

namespace QuantLib {
    class ZeroInflationTermStructure;

    template<class TS>
    class BootstrapHelper;

    typedef BootstrapHelper<ZeroInflationTermStructure>
ZeroInflationRateHelper;
}

namespace QuantLibAddin {

      class ZeroInflationRateHelper : public
ObjectHandler::LibraryObject<QuantLib::ZeroInflationRateHelper> {
      protected:
        OH_LIB_CTOR(ZeroInflationRateHelper,
QuantLib::ZeroInflationRateHelper);
    };

    class ZciisInflationHelper : public ZeroInflationRateHelper {
            ...
    };

}
```

The code in the .cpp file contains the straightforward implementation of the ZciisInflationHelper constructor.

Add these files to the QuantLibObjects project, in the TermStructures folder.

The gensrc code is written in the new file:

```
QuantLibAddin\gensrc\metadata\functions\inflationratehelper
.xml
```

We also use the category name ‘inflationratehelper’.

2. Expose the ZeroInflationTermStructure class

To expose the PiecewiseZeroInflationCurve we first create a ZeroInflationTermStructure class in QuantLibAddin.

This is only used as a base class for our PiecewiseZeroInflationCurve class and is never instantiated directly from Excel, so while we provide the following wrapper for the class we do not need gensrc code for its constructor. Create the following file:

```
QuantLibAddin\qlo\inflationtermstructure.hpp
```

The code for this file is given below:

```
#ifndef qla_zeroinflationtermstructure_hpp
#define qla_zeroinflationtermstructure_hpp

#include <qlo/termstructures.hpp>
#include <ql/termstructures/inflation/piecewisezeroinflationcurve.hpp>

namespace QuantLib {

    template<class TS>
    class BootstrapHelper;

    typedef BootstrapHelper<ZeroInflationTermStructure>
ZeroInflationRateHelper;
}

namespace QuantLibAddin {

    OH_OBJ_CLASS(ZeroInflationTermStructure, TermStructure);

} // namespace

#endif
```

Add this file to the QuantLibObjects project, in the TermStructures folder.

3. Wrap the QuantLib::PiecewiseZeroInflationCurve Class in a QuantLibAddin Class

Create the following two new files:

```
QuantLibAddin\qlo\piecewisezeroinflationcurve.hpp
QuantLibAddin\qlo\piecewisezeroinflationcurve.cpp
```

The code for the .hpp file is given below:

```
#ifndef qla_piecewisezeroinflationcurve_hpp
#define qla_piecewisezeroinflationcurve_hpp

#include <qlo/inflationtermstructure.hpp>

namespace QuantLibAddin {

    struct ZeroInflationToken {
        enum Interpolator { Linear,
                              CubicNaturalSpline };
    };

    class PiecewiseZeroInflationCurve : public
ZeroInflationTermStructure {
      public:
        PiecewiseZeroInflationCurve(
            const boost::shared_ptr<ObjectHandler::ValueObject>&
properties,
            const QuantLib::Date& referenceDate,
            const QuantLib::Calendar& calendar,
            const QuantLib::DayCounter& dayCounter,
            const QuantLib::Period& lag,
            QuantLib::Frequency frequency,
            QuantLib::Rate baseZeroRate,
            const QuantLib::Handle<QuantLib::YieldTermStructure>&
nominalTS,
            const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >&
instruments,
            QuantLib::Real accuracy,
            const std::string& interpolatorID,
            bool permanent);

        const std::vector<QuantLib::Time>&
times(ZeroInflationToken::Interpolator interpolator) const;
    };
}

#endif
```

The above is largely standard for wrapping a QuantLib class. There are three things to note. Firstly, we have an additional struct ZeroInflationToken which is a placeholder for QuantLib types of the same name. Secondly, the second last argument for the constructor is a string. This string will indicate which interpolator (normally the first template argument) is to be used in the PiecewiseZeroInflationCurve, instead of entering this as a template argument. Finally, we are going to need to write a wrapper for functions we wish to use and we give an example prototype here (the function times(...)). All of these are necessary since we are unable to expose a template class directly to Excel. Clearly more interpolation types and functions are available for the PiecewiseZeroInflationCurve, but to keep this as brief as possible we will only demonstrate these examples.

The code for the .cpp file is given below:

```
#ifdef HAVE_CONFIG_H
#include <qlo/config.hpp>
#endif

#include <qlo/piecewisezeroinflationcurve.hpp>
#include
<qlo/enumerations/factories/inflationtermstructuresfactory.hpp>
#include <ql/math/interpolations/all.hpp>

namespace QuantLibAddin {

    PiecewiseZeroInflationCurve::PiecewiseZeroInflationCurve(
            const boost::shared_ptr<ObjectHandler::ValueObject>&
properties,
            const QuantLib::Date& referenceDate,
            const QuantLib::Calendar& calendar,
            const QuantLib::DayCounter& dayCounter,
            const QuantLib::Period& lag,
            QuantLib::Frequency frequency,
            QuantLib::Rate baseZeroRate,
            const QuantLib::Handle<QuantLib::YieldTermStructure>&
nominalTS,
            const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >&
instruments,
            QuantLib::Real accuracy,
            const std::string& interpolatorID,
            bool permanent) : ZeroInflationTermStructure(properties,
permanent)
    {
        libraryObject_ = ObjectHandler::Create<boost::shared_ptr<
            QuantLib::ZeroInflationTermStructure> >()(interpolatorID,
                                                  referenceDate,
                                                  calendar,
                                                  dayCounter,
                                                  lag,
                                                  frequency,
                                                  baseZeroRate,
                                                  nominalTS,
                                                  instruments,
                                                  accuracy);

    }

    namespace InflationCall {

    class InflationCallerBase {
    public:
        virtual const std::vector<QuantLib::Time>& times(const
QuantLib::Extrapolator *extrapolator) const = 0;

        virtual ~InflationCallerBase() {}
    };

    template <class Interpolator>
    class InflationCaller : public InflationCallerBase {

        typedef QuantLib::PiecewiseZeroInflationCurve<Interpolator>
InflationCurveClass;

        const InflationCurveClass *get(const QuantLib::Extrapolator
*extrapolator) const {

            const InflationCurveClass *ret = dynamic_cast<const
InflationCurveClass*>(extrapolator);
            OH_REQUIRE(ret, "Unable to convert from type " <<
typeid(extrapolator).name()
                << " to type " << typeid(InflationCurveClass).name());
            return ret;
        }
        virtual const std::vector<QuantLib::Time>& times(const
QuantLib::Extrapolator *extrapolator) const {
            return get(extrapolator)->times();
        }
    };

    typedef ZeroInflationToken::Interpolator Token;

    std::ostream &operator<<(std::ostream &out, Token token) {

        out << "PiecewiseZeroInflationCurve<";

        switch (token) {
            case ZeroInflationToken::Linear:
                out << "Linear>";
                break;
            case ZeroInflationToken::CubicNaturalSpline:
                out << "CubicNaturalSpline>";
                break;
            default:
                OH_FAIL("Unknown value for enumeration
QuantLibAddin::ZeroInflationToken::Interpolator");
        }
        return out;
    }

    class InflationCallerFactory {

        typedef std::map<Token, InflationCallerBase*>
InflationCallerMap;
        InflationCallerMap inflationCallerMap_;

        template <class Interpolator>
        void init(Token token) {
            inflationCallerMap_[token] = new
InflationCaller<Interpolator>;
        }

    public:

        InflationCallerFactory() {

            init<QuantLib::Linear>(
Token(ZeroInflationToken::Linear));
            init<QuantLib::Cubic>(
Token(ZeroInflationToken::CubicNaturalSpline));

        }

        ~InflationCallerFactory() {
            for (InflationCallerMap::const_iterator i =
inflationCallerMap_.begin(); i != inflationCallerMap_.end(); i++)
                delete i->second;
        }

        const InflationCallerBase *getInflationCaller(Token token)
const {
            InflationCallerMap::const_iterator i =
inflationCallerMap_.find(token);
            OH_REQUIRE(i!=inflationCallerMap_.end(), "Unable to
retrieve caller for type " << token);
            return i->second;
        }

    };

    const InflationCallerFactory &inflationCallerFactory() {
        static InflationCallerFactory inflationCallerFactory_;
        return inflationCallerFactory_;
    }

    } // namespace InflationCall

#define INFLATIONCALL(FUNC) \
InflationCall::inflationCallerFactory().getInflationCaller(InflationCal
l::Token(interpolator))->FUNC(libraryObject_.get())

    const std::vector<QuantLib::Time>&
PiecewiseZeroInflationCurve::times(
        ZeroInflationToken::Interpolator interpolator) const {
        return INFLATIONCALL(times);
    }
}
```

Initially we have the constructor definition, as for other “standard” classes we expose to Excel. In this example the constructor calls the ObjectHandler::Create function which we define in the next step. Inside the namespace InflationCall we provide the logic to wrap the template class. This consists of a non-template base class

(InflationCallerBase) and a template subclass (InflationCaller). We provide a stream operator for logging and error handling. Finally we have a factory which stores a map of pointers and wrappers for member functions. See the documentation in piecewiseyieldcurve.cpp for a more detailed explanation.

Add these files to the QuantLibObjects project, in the TermStructures folder.

4. Define the ObjectHandler::Create function

Create the new file:

`QuantLibAddin\qlo\enumerations\factories\inflationtermstructuresfactory.hpp`

This file defines the Create function called by the constructor in the previous step. The code for this file is:

```
#ifndef qla_termstructuresfactory_hpp
#define qla_termstructuresfactory_hpp

#include <oh/enumerations/typefactory.hpp>
#include <ql/types.hpp>
#include <ql/termstructures/inflationtermstructure.hpp>

namespace QuantLib {
    template<class TS>
    class BootstrapHelper;

      typedef BootstrapHelper<ZeroInflationTermStructure>
ZeroInflationRateHelper;

}

namespace ObjectHandler {

    typedef
boost::shared_ptr<QuantLib::ZeroInflationTermStructure>(*ZeroInflationT
ermStructureConstructor)(
                  const QuantLib::Date& referenceDate,
            const QuantLib::Calendar& calendar,
            const QuantLib::DayCounter& dayCounter,
            const QuantLib::Period& lag,
            QuantLib::Frequency frequency,
            QuantLib::Rate baseZeroRate,
            const QuantLib::Handle<QuantLib::YieldTermStructure>&
nominalTS,
                  const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >& rh,
            QuantLib::Real accuracy);


    template<>
    class
Create<boost::shared_ptr<QuantLib::ZeroInflationTermStructure> > :
        private RegistryManager<QuantLib::ZeroInflationTermStructure,
                                EnumClassRegistry> {
    public:
        boost::shared_ptr<QuantLib::ZeroInflationTermStructure>
operator() (
                const std::string& interpolatorID,
                        const QuantLib::Date& referenceDate,
                        const QuantLib::Calendar& calendar,
                        const QuantLib::DayCounter& dayCounter,
                        const QuantLib::Period& lag,
                        QuantLib::Frequency frequency,
                        QuantLib::Rate baseZeroRate,
                        const
QuantLib::Handle<QuantLib::YieldTermStructure>& nominalTS,
                        const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >& rh,
                        QuantLib::Real accuracy) {
            ZeroInflationTermStructureConstructor
zeroInflationTermStructureConstructor =

reinterpret_cast<ZeroInflationTermStructureConstructor>(getType(interpo
latorID));
            return zeroInflationTermStructureConstructor(referenceDate,
calendar,
                                                 dayCounter, lag,
                                                 frequency,
baseZeroRate,
                                                 nominalTS, rh,
accuracy);
        }
        using RegistryManager<QuantLib::ZeroInflationTermStructure,
                              EnumClassRegistry>::registerType;
    };
 }

#endif
```

Add this file to the QuantLibObjects project, in the enumerations\factories folder.

5. Define New Types and Function Category

Open the file:

```
QuantLibAddin\gensrc\metadata\types\types.xml
```

Add the following code inside the `<DataTypes> ... </DataTypes>` tags.

```
    <DataType
defaultSuperType='enumeration'>QuantLibAddin::ZeroInflationToken::Inter
polator</DataType>
    <DataType
defaultSuperType='libraryClass'>QuantLib::ZeroInflationRateHelper</Data
Type>
    <DataType
defaultSuperType='objectClass'>QuantLibAddin::ZeroInflationRateHelper</
DataType>
    <DataType
defaultSuperType='objectClass'>QuantLibAddin::PiecewiseZeroInflationCur
ve</DataType>
    <DataType
defaultSuperType='libraryTermStructure'>QuantLib::ZeroInflationTermStru
cture</DataType>
```

Open the file:

`QuantLibAddin\gensrc\config\categories.xml`

Add the category piecewisezeroinflaitoncurve:

```
<root>

  <addinCategoryNames>
    <categoryName>abcd</categoryName>
    <categoryName>accountingengines</categoryName>
    ...
    <categoryName>piecewiseyieldcurve</categoryName>
    <categoryName>piecewisezeroinflaitoncurve</categoryName>
    <categoryName>prices</categoryName>
    ...
    <categoryName>volatility</categoryName>
    <categoryName>quotes</categoryName>
  </addinCategoryNames>

</root>
```

Create the file:

`QuantLibAddin\gensrc\metadata\functions\piecewisezeroinflaitoncurve.xml`

The code for this file is given below:

```
<Category name='piecewisezeroinflationcurve'>
    <description>functions to construct and use
PiecewiseZeroInflationCurve objects.</description>
    <displayName>Piecewise Zero Inflation Curves</displayName>
    <xlFunctionWizardCategory>QuantLib -
Financial</xlFunctionWizardCategory>
    <serializationIncludes>

<include>qlo/enumerations/factories/inflationtermstructuresfactory.hpp<
/include>
        <include>qlo/inflationratehelper.hpp</include>
        <include>qlo/piecewisezeroinflationcurve.hpp</include>
        <include>qlo/yieldtermstructures.hpp</include>

<include>ql/termstructures/inflation/piecewisezeroinflationcurve.hpp</i
nclude>
    </serializationIncludes>
    <addinIncludes>
        <include>qlo/yieldtermstructures.hpp</include>
        <include>qlo/inflationratehelper.hpp</include>
        <include>qlo/piecewisezeroinflationcurve.hpp</include>
    </addinIncludes>
    <copyright>
    </copyright>
    <Functions>

        <!-- PiecewiseZeroInflationCurve constructor -->

        <Constructor name='qlPiecewiseZeroInflationCurve'>

<libraryFunction>PiecewiseZeroInflationCurve</libraryFunction>
            <SupportedPlatforms>
                <!--SupportedPlatform name='Excel'
calcInWizard='false'/-->
                <SupportedPlatform name='Excel'/>
                <SupportedPlatform name='Cpp'/>
            </SupportedPlatforms>
            <ParameterList>
                <Parameters>
                    <Parameter name='ReferenceDate'>
                      <type>QuantLib::Date</type>
                      <tensorRank>scalar</tensorRank>
                      <description>pricing reference
date.</description>
                    </Parameter>
                    <Parameter name='Calendar'>
                      <type>QuantLib::Calendar</type>
                      <tensorRank>scalar</tensorRank>
                      <description>holiday calendar (e.g. TARGET) to
advance from global EvaluationDate.</description>
                    </Parameter>
                    <Parameter name='DayCounter' default='"Actual/365
(Fixed)"'>
                      <type>QuantLib::DayCounter</type>
                      <tensorRank>scalar</tensorRank>
                      <description>DayCounter ID.</description>
                    </Parameter>
                    <Parameter name='Lag'>
                      <type>QuantLib::Period</type>
                      <tensorRank>scalar</tensorRank>
                      <description>lag period.</description>
                    </Parameter>
                    <Parameter name='Frequency'>
                      <type>QuantLib::Frequency</type>
                      <tensorRank>scalar</tensorRank>
                      <description>frequency.</description>
                    </Parameter>
                    <Parameter name='BaseZeroRate'>
                        <type>QuantLib::Rate</type>
                        <tensorRank>scalar</tensorRank>
                        <description>base zero rate.</description>
                    </Parameter>
                    <Parameter name='YieldCurve'>
                      <type>QuantLib::YieldTermStructure</type>
                      <superType>libToHandle</superType>
                      <tensorRank>scalar</tensorRank>
                      <description>yield term structure.</description>
                    </Parameter>
                    <Parameter name='InflationRateHelpers'>
                        <type>QuantLib::ZeroInflationRateHelper</type>
                        <tensorRank>vector</tensorRank>
                        <description>vector of rate-
helpers.</description>
                    </Parameter>
                    <Parameter name='Accuracy' default='1.0e-12'>
                        <type>QuantLib::Real</type>
                        <tensorRank>scalar</tensorRank>
                        <description>Bootstrapping
accuracy.</description>
                    </Parameter>
                    <Parameter name='InterpolatorID'
default='"LogLinear"'>
                        <type>string</type>
                        <tensorRank>scalar</tensorRank>
                        <description>BackwardFlat, ForwardFlat, Linear,
LogLinear, CubicSpline, or LogCubic.</description>
                    </Parameter>
                </Parameters>
            </ParameterList>
        </Constructor>

        <!-- PiecewiseYieldCurve interface -->

        <Member name='qlPiecewiseZeroInflationCurveTimes'
type='QuantLibAddin::PiecewiseZeroInflationCurve'
superType='objectClass'>
            <description>Retrieve list of Times for the given
PiecewiseZeroInflationCurve&lt;Interpolator&gt;.</description>
            <libraryFunction>times</libraryFunction>
            <SupportedPlatforms>
                <SupportedPlatform name='Excel'/>
            </SupportedPlatforms>
            <ParameterList>
                <Parameters>
                    <Parameter name='Interpolator'>

<type>QuantLibAddin::ZeroInflationToken::Interpolator</type>
                        <tensorRank>scalar</tensorRank>
                        <description>interpolator.</description>
                    </Parameter>
                </Parameters>
            </ParameterList>
            <ReturnValue>
                <type>QuantLib::Time</type>
                <tensorRank>vector</tensorRank>
            </ReturnValue>
        </Member>
    </Functions>

</Category>
```

Add the file to the qlgensrc project, in the functions folder.

6. Define the Constructors for the Enumerated Classes

Open the file:

`QuantLibAddin\qlo\enumerations\constructors\enumeratedclass es.hpp`

Add the line:

```
#include
<qlo/enumerations/factories/inflationtermstructuresfactory.hpp>
```

Inside the QuantLibAddin namespace, add the code:

```
      /* *** InflationTermStructures *** */
      /* *** PiecewiseZeroInflationCurve *** */
    boost::shared_ptr<QuantLib::ZeroInflationTermStructure>
LINEAR_PiecewiseZeroInflationCurve(
                  const QuantLib::Date& referenceDate,
            const QuantLib::Calendar& calendar,
            const QuantLib::DayCounter& dayCounter,
            const QuantLib::Period& lag,
            QuantLib::Frequency frequency,
            QuantLib::Rate baseZeroRate,
                  const QuantLib::Handle<QuantLib::YieldTermStructure>&
nominalTS,
                  const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >&
instruments,
            QuantLib::Real accuracy);
    boost::shared_ptr<QuantLib::ZeroInflationTermStructure>
CUBICNATURALSPLINE_PiecewiseZeroInflationCurve(
                  const QuantLib::Date& referenceDate,
            const QuantLib::Calendar& calendar,
            const QuantLib::DayCounter& dayCounter,
            const QuantLib::Period& lag,
            QuantLib::Frequency frequency,
            QuantLib::Rate baseZeroRate,
                  const QuantLib::Handle<QuantLib::YieldTermStructure>&
nominalTS,
                  const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >&
instruments,
            QuantLib::Real accuracy);
```

Open the file:

QuantLibAddin\qlo\enumerations\constructors\enumeratedclasses.cpp

Add the line:

```
#include <ql/termstructures/inflation/piecewisezeroinflationcurve.hpp>
```

Inside the QuantLibAddin namespace, add the code:

```
      /* *** InflationTermStructures *** */
      /* *** PiecewiseZeroInflationCurve *** */
    boost::shared_ptr<QuantLib::ZeroInflationTermStructure>
LINEAR_PiecewiseZeroInflationCurve(
                  const QuantLib::Date& referenceDate,
            const QuantLib::Calendar& calendar,
            const QuantLib::DayCounter& dayCounter,
            const QuantLib::Period& lag,
            QuantLib::Frequency frequency,
            QuantLib::Rate baseZeroRate,
                  const QuantLib::Handle<QuantLib::YieldTermStructure>&
nominalTS,
                  const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >&
instruments,
            QuantLib::Real accuracy) {
            return
boost::shared_ptr<QuantLib::ZeroInflationTermStructure>(new

      QuantLib::PiecewiseZeroInflationCurve<QuantLib::Linear>(
                                                      referenceDate,
                                                      calendar,
                                                      dayCounter,
                                                      lag,
                                                      frequency,
                                                      baseZeroRate,
                                                      nominalTS,
                                                      instruments,
                                                      accuracy));
      }
    boost::shared_ptr<QuantLib::ZeroInflationTermStructure>
CUBICNATURALSPLINE_PiecewiseZeroInflationCurve(
                  const QuantLib::Date& referenceDate,
            const QuantLib::Calendar& calendar,
            const QuantLib::DayCounter& dayCounter,
            const QuantLib::Period& lag,
            QuantLib::Frequency frequency,
            QuantLib::Rate baseZeroRate,
                  const QuantLib::Handle<QuantLib::YieldTermStructure>&
nominalTS,
                  const
std::vector<boost::shared_ptr<QuantLib::ZeroInflationRateHelper> >&
instruments,
            QuantLib::Real accuracy) {
            return
boost::shared_ptr<QuantLib::ZeroInflationTermStructure>(new

      QuantLib::PiecewiseZeroInflationCurve<QuantLib::Cubic>(
                                                      referenceDate,
                                                      calendar,
                                                      dayCounter,
                                                      lag,
                                                      frequency,
                                                      baseZeroRate,
                                                      nominalTS,
                                                      instruments,
                                                      accuracy,
QuantLib::Cubic(QuantLib::CubicInterpolation::Spline, false,
QuantLib::CubicInterpolation::SecondDerivative, 0.0,
QuantLib::CubicInterpolation::SecondDerivative, 0.0)));
      }
```

Finally, open the file:

`QuantLibAddin\gensrc\metadata\enumerations\enumeratedclass.xml`

Inside the `<EnumeratedClassGroups>`... `</EnumeratedClassGroups>` tags, add the code:

```
    <EnumeratedClassGroup class='QuantLib::ZeroInflationTermStructure'>
     <!--
<includeFile>qlo/enumerations/factories/interpolationsfactory.hpp</incl
udeFile> -->
      <EnumeratedClasses>
        <EnumeratedClass>
          <string>Linear</string>
          <value>LINEAR_PiecewiseZeroInflationCurve</value>
          <libraryClass>QuantLib::LinearInterpolation</libraryClass>
        </EnumeratedClass>
        <EnumeratedClass>
          <string>CubicNaturalSpline</string>
          <value>CUBICNATURALSPLINE_PiecewiseZeroInflationCurve</value>
          <libraryClass>QuantLib::CubicNaturalSpline</libraryClass>
        </EnumeratedClass>
      </EnumeratedClasses>
    </EnumeratedClassGroup>
```

7. Auto-generate gensrc Files

As in the stock tutorial, we need to run gensrc to auto generate the new source code. Build project qlgensrc. This auto-generates the following eleven new files (some output omitted):

```
1>c:\..\QuantLibXL\qlxl/register/register_piecewisezeroinflationcurve.cpp:
created
1>c:\..\QuantLibXL\qlxl/functions/piecewisezeroinflationcurve.cpp:
created
1>c:\..\QuantLibAddin\Addins\Cpp/piecewisezeroinflationcurve.cpp:
created
1>c:\..\QuantLibAddin\Addins\Cpp/piecewisezeroinflationcurve.hpp:
created
1>c:\..\QuantLibAddin\qlo\valueobjects/vo_piecewisezeroinflationcurve.hpp:
created
1>c:\..\QuantLibAddin\qlo\valueobjects/vo_piecewisezeroinflationcurve.cpp:
created
1>c:\..\QuantLibAddin\qlo\serialization/create/create_piecewisezeroinflationcurve.hpp:
created
1>c:\..\QuantLibAddin\qlo\serialization/create/create_piecewisezeroinflationcurve.cpp:
created
1>c:\..\QuantLibAddin\qlo\serialization/register/serialization_piecewisezeroinflationcurv
e.hpp:    created
1>c:\..\QuantLibAddin\qlo\serialization/register/serialization_piecewisezeroinflationcurv
e.cpp:    created
1>c:\..\QuantLibAddin\Docs\auto.pages/piecewisezeroinflationcurve.docs:
created
```

We can ignore the .docs file and the two files created in QuantLibAddin/Addins. The remaining files need to be added to the relevant projects as given by the following table:

Add files(s) …	**To project …**	**In folder …**
QuantLibXL\qlxl/register/register_piecewisezeroinflationcurve.cpp	QuantLib XLStatic	register
QuantLibXL\qlxl/functions/piecewisezeroinflationcurve.cpp	QuantLib XLStatic	functions
QuantLibAddin\qlo\valueobjects/vo_piecewisezeroinflationcurve.*pp	QuantLib Objects	valueobjects
QuantLibAddin\qlo\serialization/create/create_piecewisezeroinflationcurve.*pp	QuantLib Objects	serialization/create
QuantLibAddin\qlo\serialization/register/serialization_piecewisezeroinflationcurve.*pp	QuantLib Objects	serialization/register

8. Add ZeroInflationTermStructure interface

Open the file:

`QuantLibAddin\gensrc\metadata\functions\termstructures.xml`

Add the following code between `<Functions>` and `</Functions>` tags:

```
    <Member name='qlZeroInflationTSZeroRate'
type='QuantLib::ZeroInflationTermStructure'
superType='libraryTermStructure' loopParameter='Dates'>
      <description>zero rate for a given ZeroInflationTermStructure
object.</description>
      <libraryFunction>zeroRate</libraryFunction>
      <SupportedPlatforms>
        <SupportedPlatform name='Excel'/>
      </SupportedPlatforms>
      <ParameterList>
        <Parameters>
          <Parameter name='Dates' exampleValue ="'37413,39248">
            <type>QuantLib::Date</type>
            <tensorRank>vector</tensorRank>
            <description>vector of dates.</description>
          </Parameter>
          <Parameter name='AllowExtrapolation' const='False'
default='false'>
            <type>bool</type>
            <tensorRank>scalar</tensorRank>
            <description>TRUE allows extrapolation.</description>
          </Parameter>
        </Parameters>
      </ParameterList>
      <ReturnValue>
        <type>QuantLib::Probability</type>
        <tensorRank>vector</tensorRank>
      </ReturnValue>
    </Member>
```

Note that we have used a loop variable, see http://quantlib.org/quantlibxl/loops.html for a brief explanation.

In addition, between both the `<serializationIncludes> ... </serializationIncludes>` and the `<addinIncludes> ... </addinIncludes>` tags add the line:

```
<include>qlo/inflationtermstructure.hpp</include>
```