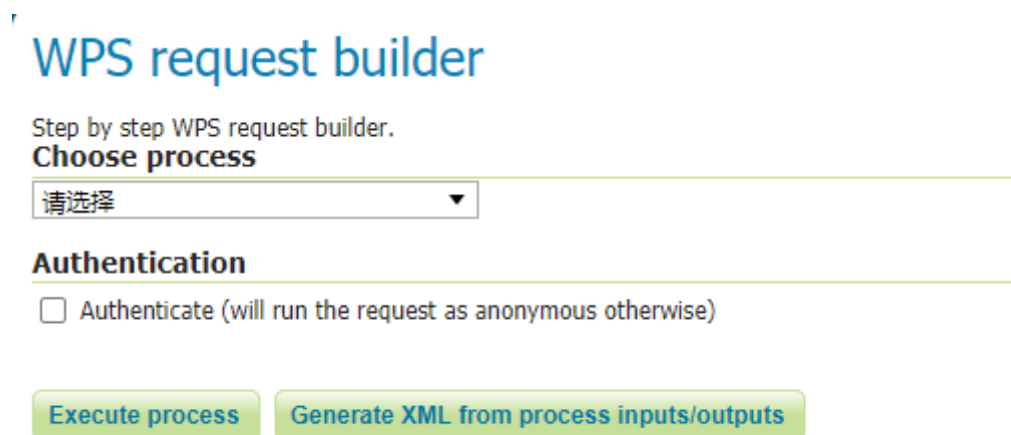


WPS演示

wps请求构造器

geoserver wps扩展包含一个请求生成器，用于通过Web管理界面。该工具还可以用于演示流程，并构建自己的示例。

WPS请求生成器主要由一个列出所有可用进程的选择框和两个按钮组成，一个用于提交WPS请求，另一个用于显示POST请求的外观。



The screenshot shows the 'WPS request builder' web interface. It has a title 'WPS request builder' in blue. Below it is a subtitle 'Step by step WPS request builder.' and a section header 'Choose process'. There is a dropdown menu with the text '请选择' (Please select). Below this is a section header 'Authentication' with a checkbox labeled 'Authenticate (will run the request as anonymous otherwise)'. At the bottom, there are two green buttons: 'Execute process' and 'Generate XML from process inputs/outputs'.

根据所选的过程和输入，显示会发生变化。JTS过程可以作为任何基于GML/WKT的特性集合、URL引用或子过程的输入。特定于geoserver的进程具有所有这些选项，还包括选择geoserver层作为输入的能力。

对于每个流程，将根据与该流程相关联的必需和可选参数（如果有）显示一个表单。要将该进程视为一个POST请求，请单击 **Generate XML from process inputs/outputs** 按钮。要执行该进程，请单击 **Execute Process** 按钮。响应将显示在窗口中。

进程

- JTS:area 返回几何图形的面积，以几何图形的单位为单位。假设为笛卡尔平面，因此只建议对非地理CRSes使用此过程。
- JTS:boundary 返回一个几何边界。对于多边形，返回与多边形边界相等的线性环或多线串。对于linestring，返回一个与linestring的端点相等的多点。对于点，返回一个空的几何集合。
- JTS:buffer 返回一个多边形几何图形，表示在其外部周围扩大了给定距离的输入几何图形。
- JTS:centroid 返回几何图形的几何形心。输出为单点。质心点可以位于几何图形之外
- JTS:contains 测试是否第二几何没有点位于第一几何的外部，而第二几何的内部至少有一个点位于第一几何的内部
- JTS:convexHull 返回包含整个输入几何图形的最小凸多边形
- JTS:crosses 测试两个几何图形是否有一些，但不是全部的内部共同点
- JTS:densify 返回一个空间上等价的几何图形，其中添加顶点以确保线段不超过给定的距离
- JTS:difference 返回一个几何图形，表示在一个几何图形中包含但在另一个几何图形中不包含的点。其结果可能是一个异构的几何集合
- JTS:dimension 返回几何或几何集合的最大维数:点为0，线为1，多边形为2
- JTS:disjoint 测试两个几何图形是否没有任何共同点
- JTS:distance 返回两个几何图形之间的最小距离。测量以输入单位表示，所以不推荐使用地理坐标
- JTS:endpoint 返回一个点几何图形，它等于一个LineString的最后一个顶点
- JTS:envelope 返回包含几何图形的最小边界框多边形。对于点几何，返回相同的点
- JTS>equalsExact 测试两个几何图形在顶点对顶点的基础上是否相同

- `JTS:equalsExactTolerance` 测试两个几何图形在顶点对顶点的基础上是否相同，直到顶点距离公差
- `JTS:exteriorRing` 返回多边形几何的外环
- `JTS:geometryType` 返回几何图形类型的名称。值是一个点，线串，多边形，多点，多线串，多多边形，几何集合
- `JTS:getGeometryN` 返回几何集合中给定索引处的几何元素。索引从0开始
- `JTS:getX` 返回点几何图形的X值(第一纵坐标)。对于其他几何类型，返回形心的X值
- `JTS:getY` 返回点几何图形的Y值(第二纵坐标)。对于其他几何类型，返回形心的Y值
- `JTS:interiorPoint` 返回位于几何图形内部(如果可能的话)或位于其边界上的点
- `JTS:interiorRingN` 从包含由给定索引确定的内环(孔)的多边形返回线性环。第一个内环的指数是0。如果没有内环，返回null
- `JTS:intersection` 返回一个几何图形，表示两个几何图形的共同点。其结果可能是一个异构的几何集合。如果没有交集，返回一个空几何图形
- `JTS:intersects` 测试两个几何图形是否相交
- `JTS:isClosed` 测试线性几何中的初始顶点是否等于最终顶点。点和多边形总是返回True
- `JTS:isEmpty` 测试几何图形是否不包含顶点
- `JTS:isRing` 测试一个几何图形是否既封闭又简单
- `JTS:isSimple` 测试一个几何图形是否拓扑简单。点、多边形、闭线弦和线环总是很简单的。如果没有两个相同的点，其他几何图形被认为是简单的
- `JTS:isValid` 测试一个几何图形是否拓扑有效
- `JTS:isWithinDistance` 测试两个几何图形之间的最小距离是否小于一个公差值
- `JTS:length` 返回几何图形中所有线段的总长度。测量是用源单位给出的，所以不推荐使用地理坐标
- `JTS:numGeometries` 返回几何集合中元素的总数。如果不是几何集合，返回1。如果为空，返回0
- `JTS:numInteriorRing` 返回多边形几何中内环的总数。点和线返回0
- `JTS:numPoints` 返回给定几何图形中的顶点数
- `JTS:overlaps` 测试两个几何图形是否共享一些但不是所有的内部点。点或线总是返回False
- `JTS:pointN` 返回一个点几何图形，等于由给定索引确定的几何图形中的第n个顶点。第一个顶点的索引是0
- `JTS:polygonize` 根据线串的轮廓创建一组多边形。行字符串必须正确的节点(例如，只接触端点)
- `JTS:relate` 返回用于输入几何图形之间的空间关系的DE-9IM交叉矩阵字符串。矩阵弦的形式为`[II][IB][IE][BI][BB][BE][EI][EB][EE]`，其中I=内部，B=边界，E=外部。矩阵符号是2,1,0或F
- `JTS:relatePattern` 测试两个几何图形之间的空间关系是否与给定的DE-9IM交叉矩阵模式匹配。模式以`[II][IB][IE][BI][BB][BE][EI][EB][EE]`的形式给出，其中I=内部，B=边界，E=外部。图案符号可以是2、1、0、F或*
- `JTS:reproject` 将给定的几何图形重新投影到提供的坐标参考系统中
- `JTS:simplify` 返回一个根据Douglas-Peucker算法简化的几何图形(在顶点中减少)
- `JTS:splitPolygon` 通过linestring分割多边形
- `JTS:startPoint` 返回一个点几何图形，等于LineString的第一个顶点
- `JTS:symDifference` 返回一个几何图形，表示两个几何图形中包含但不同时包含的点。其结果可能是一个异构的几何集合
- `JTS:touches` 测试两个几何图形是否至少有一个共同的边界点，但没有共享的内部点
- `JTS:union` 返回表示几何集合中任何几何图形中包含的所有点的几何图形
- `JTS:within` 测试第一个几何图形是否包含在第二个几何图形中
- `geo:area` 返回几何图形的面积，以几何图形的单位为单位。假设为笛卡尔平面，因此只建议对非地理CRSes使用此过程。
- `geo:boundary` 返回一个几何边界。对于多边形，返回与多边形边界相等的线性环或多线串。对于linestring，返回一个与linestring的端点相等的多点。对于点，返回一个空的几何集合。
- `geo:buffer` 返回一个多边形几何图形，表示在其外部周围扩大了给定距离的输入几何图形。
- `geo:centroid` 返回几何图形的几何形心。输出为单点。质心点可以位于几何图形之外
- `geo:contains` 测试是否第二几何没有点位于第一几何的外部，而第二几何的内部至少有一个点位于第一几何的内部
- `geo:convexHull` 返回包含整个输入几何图形的最小凸多边形
- `geo:crosses` 测试两个几何图形是否有一些，但不是全部的内部共同点

- geo:densify 返回一个空间上等价的几何图形，其中添加顶点以确保线段不超过给定的距离
- geo:difference 返回一个几何图形，表示在一个几何图形中包含但在另一个几何图形中不包含的点。其结果可能是一个异构的几何集合
- geo:dimension 返回几何或几何集合的最大维数:点为0，线为1，多边形为2
- geo:disjoint 测试两个几何图形是否没有任何共同点
- geo:distance 返回两个几何图形之间的最小距离。测量以输入单位表示，所以不推荐使用地理坐标
- geo:endpoint 返回一个点几何图形，它等于一个LineString的最后一个顶点
- geo:envelope 返回包含几何图形的最小边界框多边形。对于点几何，返回相同的点
- geo:equalsExact 测试两个几何图形在顶点对顶点的基础上是否相同
- geo:equalsExacttolerance 测试两个几何图形在顶点对顶点的基础上是否相同，直到顶点距离公差
- geo:exteriorRing 返回多边形几何的外环
- geo:geometryType 返回几何图形类型的名称。值是一个点，线串，多边形，多点，多线串，多多边形，几何集合
- geo:getGeometryN 返回几何集合中给定索引处的几何元素。索引从0开始
- geo:getX 返回点几何图形的X值(第一纵坐标)。对于其他几何类型，返回形心的X值
- geo:getY 返回点几何图形的Y值(第二纵坐标)。对于其他几何类型，返回形心的Y值
- geo:interiorPoint 返回位于几何图形内部(如果可能的话)或位于其边界上的点
- geo:interiorRingN 从包含由给定索引确定的内环(孔)的多边形返回线性环。第一个内环的指数是0。如果没有内环，返回null
- geo:intersection 返回一个几何图形，表示两个几何图形的共同点。其结果可能是一个异构的几何集合。如果没有交集，返回一个空几何图形
- geo:intersects 测试两个几何图形是否相交
- geo:isClosed 测试线性几何中的初始顶点是否等于最终顶点。点和多边形总是返回True
- geo:isEmpty 测试几何图形是否不包含顶点
- geo:isRing 测试一个几何图形是否既封闭又简单
- geo:isSample 测试一个几何图形是否拓扑简单。点、多边形、闭线弦和线环总是很简单的。如果没有两个相同的点，其他几何图形被认为是简单的
- geo:isValid 测试一个几何图形是否拓扑有效
- geo:isWithinDistance 测试两个几何图形之间的最小距离是否小于一个公差值
- geo:length 返回几何图形中所有线段的总长度。测量是用源单位给出的，所以不推荐使用地理坐标
- geo:numGeometries 返回几何集合中元素的总数。如果不是几何集合，返回1。如果为空，返回0
- geo:numInteriorRing 返回多边形几何中内环的总数。点和线返回0
- geo:numPoints 返回给定几何图形中的顶点数
- geo:overlaps 测试两个几何图形是否共享一些但不是所有的内部点。点或线总是返回False
- geo:pointN 返回一个点几何图形，等于由给定索引确定的几何图形中的第n个顶点。第一个顶点的索引是0
- geo:polygonize 根据线串的轮廓创建一组多边形。行字符串必须正确的节点(例如，只接触端点)
- geo:relate 返回用于输入几何图形之间的空间关系的DE-9IM交叉矩阵字符串。矩阵弦的形式为[I][I][IB][IE][BI][BB][BE][EI][EB][EE]，其中I=内部，B=边界，E=外部。矩阵符号是2,1,0或F
- geo:relatePattern 测试两个几何图形之间的空间关系是否与给定的DE-9IM交叉矩阵模式匹配。模式以[I][I][IB][IE][BI][BB][BE][EI][EB][EE]的形式给出，其中I=内部，B=边界，E=外部。图案符号可以是2、1、0、F或*
- geo:reproject 将给定的几何图形重新投影到提供的坐标参考系统中
- geo:simplify 返回一个根据Douglas-Peucker算法简化的几何图形(在顶点中减少)
- geo:splitPolygon 通过linestring分割多边形
- geo:startPoint 返回一个点几何图形，等于LineString的第一个顶点
- geo:symDifference 返回一个几何图形，表示两个几何图形中包含但不同时包含的点。其结果可能是一个异构的几何集合
- geo:touches 测试两个几何图形是否至少有一个共同的边界点，但没有共享的内部点
- geo:union 返回表示几何集合中任何几何图形中包含的所有点的几何图形

- `geo:within` 测试第一个几何图形是否包含在第二个几何图形中
- `gs:AddCoverages` 返回由两个源光栅逐像素相加生成的光栅。源光栅必须有相同的边框和分辨率
- `gs:Aggregate` 在一个特性属性上计算一个或多个聚合函数。函数包括Count、Average、Max、中值、Min、StdDev和Sum
- `gs:AreaGrid` 计算给定地理范围的栅格网格，单元格值等于该单元格在地球表面所表示的面积。面积是使用EckertIV投影计算的
- `gs:BarnesSurface` 使用巴恩斯分析计算在一组不规则数据点上的插值表面
- `gs:Bounds` 计算输入特征的边界框
- `gs:BufferFeatureCollection` 通过作为参数或由特性属性提供的距离值缓冲特性。基于笛卡尔距离计算缓冲区
- `gs:Centroid` 计算特征的几何中心
- `gs:Clip` 剪辑特征到一个给定的几何
- `gs:CollectGeometries` 收集输入特性的默认几何图形，并将它们组合到单个几何图形集合中
- `gs:Contour` 为光栅中的值在指定的间隔或水平上计算等高线
- `gs:Count` 计算特征集合中特征的数量
- `gs:CropCoverage` 返回光栅中被给定几何图形限定的部分
- `gs:Feature` 将几何图形转换为功能集合
- `gs:GeorectifyCoverage` 通过地面控制点使用gdal_warp对光栅进行地理校正
- `gs:GetFullCoverage` 返回目录中的栅格，带有可选的筛选
- `gs:Grid` 生成具有地理标定的单元格规则网格。
- `gs:Heatmap` 计算热图表面在一组数据点和输出作为一个单波段光栅
- `gs:Import` 将特性集合导入目录
- `gs:InclusionFeatureCollection` 返回一个特征集合，该特征集合由第一个特征集合的特征组成，该特征集合在空间上包含在第二个特征集合的至少一个特征中
- `gs:IntersectionFeatureCollection` 两个特征集合的空间交集，包括合并两个特征集合的属性
- `gs:LRSGeocode` 从LRS特征中提取给定测量值下的点
- `gs:LRSMeasure` 沿着特性(属性为lrs_measure的特性)计算点的度量。该点是沿着最近的特征测量的
- `gs:LRSegment` 从LRS特征中提取给定开始和结束测度之间的部分
- `gs:MultiplyCoverages` 返回由两个源光栅逐像素相乘生成的光栅。源光栅必须有相同的边框和分辨率
- `gs:Nearest` 返回给定特性集合中到给定点的距离最小的特性
- `gs:PagedUnique` 获取指定字段上给定featurecall的惟一值列表，允许可选分页
- `gs:PointBuffers` 返回具有以给定点为中心的指定半径的圆形缓冲区多边形的集合
- `gs:PointStacker` 将网格上的点集合聚合为每个网格单元中的一个点
- `gs:PolygonExtraction` 根据相等或在给定范围内的区域，从光栅中提取矢量多边形
- `gs:Query` 使用可选筛选器和要包含的可选属性列表查询特性集合。也可用于转换特性集合格式
- `gs:RangeLookup` 将连续光栅重新分类为由一组范围定义的整数值
- `gs:RasterAsPointCollection` 返回光栅像素的点特征集合。带值作为属性提供
- `gs:RasterZonalStatistics` 计算在一组多边形区域中一定数量的分布的统计量
- `gs:RectangularClip` 裁剪特征到指定的矩形范围
- `gs:Reproject` 重新项目功能到一个提供的坐标参考系统。还可以强制功能集合具有给定的CRS
- `gs:ReprojectGeometry` 将给定的几何图形重新投影到提供的坐标参考系统中
- `gs:ScaleCoverage` 返回给定光栅的缩放和转换版本
- `gs:Simplify` 通过使用Douglas-Peucker简化减少顶点来简化特征几何
- `gs:Snap` 返回最接近给定点的特性集合中的特性。添加了距离和方位的属性
- `gs:StoreCoverage` 在服务器上存储光栅
- `gs:StyleCoverage` 使用给定的SLD和光栅符号来样式光栅
- `gs:Transform` 通过重命名、删除和计算新属性从输入特性集合计算新特性集合。属性值被指定为形式name=expression中的ECQL表达式
- `gs:UnionFeatureCollection` 返回包含来自两个输入特性集合的所有特性的单个特性集合。输出属性模式是来自输入的属性的组合。名称相同但类型不同的属性将被转换为字符串

- gs:Unique 返回特性集合中给定属性的唯一值
- gs:VectorZonalStatistics 计算给定属性在一组多边形区域中的分布统计信息。输入必须是点
- gt:VectorToRaster 使用属性指定单元格值，将部分或全部特性集合转换为栅格网格
- polygonlablprocess:PolyLabeller 计算可达极点，多边形中最远的内点
- ras:AddCoverages 返回由两个源光栅逐像素相加生成的光栅。源光栅必须有相同的边框和分辨率
- ras:Affine 返回输入栅格上的仿射变换的结果
- ras:AreaGrid 计算给定地理范围的栅格网格，单元格值等于该单元格在地球表面所表示的面积。面积是使用EckertIV投影计算的
- ras:BandMerge 返回由合并输入光栅带所生成的光栅。源光栅必须具有相同的CRS
- ras:BandSelect 返回由从输入光栅中选择某些波段生成的光栅
- ras:Contour 为光栅中的值在指定的间隔或水平上计算等高线
- ras:CovarianceClassStats 从分类到箱子/类的覆盖率值计算统计数据
- ras:CropCoverage 返回光栅中被给定几何图形限定的部分
- ras:jiffle 由jiffle提供的地图代数
- ras:MultiplyCoverages 返回由两个源光栅逐像素相乘生成的光栅。源光栅必须有相同的边框和分辨率
- ras:NormalizeCoverage 用值除以最大值来标准化覆盖范围
- ras:PolygonExtraction 根据相等或在给定范围内的区域，从光栅中提取矢量多边形
- ras:RangeLookup 将连续光栅重新分类为由一组范围定义的整数值
- ras:RasterAsPointCollection 返回光栅像素的点特征集合。带值作为属性提供
- ras:RasterZonalStatics 计算在一组多边形区域中一定数量的分布的统计量
- ras:ScaleCoverage 返回给定光栅的缩放和转换版本
- ras:StyleCoverage 使用给定的SLD和光栅符号来样式光栅
- ras:TransparencyFill 填满透明像素
- vec:Aggregate 在一个特性属性上计算一个或多个聚合函数。函数包括Count、Average、Max、中值、Min、StdDev和Sum
- vec:BarnesSurface 使用巴恩斯分析计算在一组不规则数据点上的插值表面
- vec:Bounds 计算输入特征的边界框
- vec:BufferFeatureCollection 通过作为参数或由特性属性提供的距离值缓冲特性。基于笛卡尔距离计算缓冲区
- vec:Centroid 计算特征的几何中心
- vec:ClassifyByRange 计算一个新属性以按向量数据集上的间隔对另一个属性进行分类
- vec:Clip 截取特征到一个给定的几何
- vec:CollectGeometries 收集输入特性的默认几何图形，并将它们组合到单个几何图形集合中
- vec:Count 计算特征集合中特征的数量
- vec:Feature 将几何图形转换为功能集合
- vec:FeatureClassStats 根据分类到箱子/类的特性值计算统计信息
- vec:Grid 生成具有地理标定的单元格规则网格
- vec:Heatmap 计算热图表面在一组数据点和输出作为一个单波段光栅
- vec:InclusionFeatureCollection 返回一个特征集合，该特征集合由第一个特征集合的特征组成，该特征集合在空间上包含在第二个特征集合的至少一个特征中
- vec:IntersectionFeatureCollection 两个特征集合的空间交集，包括合并两个特征集合的属性
- vec:LRSGeocode 从LRS特征中提取给定测量值下的点
- vec:LRSMeasure 沿着特性(属性为lrs_measure的特性)计算点的度量。该点是沿着最近的特征测量的
- vec:LRSegment 从LRS特征中提取给定开始和结束测度之间的部分
- vec:MultiplyCoverages 返回由两个源光栅逐像素相乘生成的光栅。源光栅必须有相同的边框和分辨率
- vec:Nearest 返回给定特性集合中到给定点的距离最小的特性
- vec:PointBuffers 返回具有以给定点为中心的指定半径的圆形缓冲区多边形的集合
- vec:PointStacker 将网格上的点集合聚合为每个网格单元中的一个点
- vec:Query 使用可选筛选器和要包含的可选属性列表查询特性集合。还可以用来转换功能集合格式

- vec:RectangularClip 截取特征到一个矩形范围
- vec:Reproject 将要素重新投影到提供的坐标参考系统中。也可以强制要素集合具有给定的CRS
- vec:Simplify 通过使用Douglas-Peucker简化减少顶点来简化特征几何
- vec:Snap 返回最接近给定点的特性集合中的特性。添加了距离和方位的属性
- vec:Transform 通过重命名、删除和计算新属性从输入特性集合计算新特性集合。属性值被指定为形式name=expression中的ECQL表达式
- vec:UnionFeatureCollection 返回包含来自两个输入特性集合的所有特性的单个特性集合。输出属性模式是来自输入的属性的组合。名称相同但类型不同的属性将被转换为字符串
- vec:Unique 返回特性集合中给定属性的唯一值
- vec:VectorToRaster 使用属性指定单元格值，将部分或全部特性集合转换为栅格网格
- vec:VectorZonaStatics 计算给定属性在一组多边形区域中的分布统计信息。输入必须是点

WPS试用

1、确定toop:states边界

WPS request builder

Step by step WPS request builder.

Choose process

gs:Bounds

Computes the bounding box of the input features. ([WPS DescribeProcess](#))

Process inputs

features* - FeatureCollection

Input feature collection

VECTOR_LAYER
toop:states

Process outputs

bounds* - ReferencedEnvelope

Bounding box of input features

☒ Generate

Authentication

☐ Authenticate (will run the request as anonymous otherwise)

Execute process Generate XML from process inputs/outputs

-124.73142200000001 24.955967-66.969849 49.371735

```
<?xml version="1.0" encoding="UTF-8"?><wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
  <ows:Identifier>gs:Bounds</ows:Identifier>
  <wps>DataInputs>
    <wps:Input>
      <ows:Identifier>features</ows:Identifier>
      <wps:Reference mimeType="text/xml" xlink:href="http://geoserver/wfs"
method="POST">
        <wps:Body>
          <wfs:GetFeature service="WFS" version="1.0.0" outputFormat="GML2"
xmlns:topp="http://www.openplans.org/topp">
            <wfs:Query typeName="topp:states"/>
          </wfs:GetFeature>
        </wps:Body>
      </wps:Reference>
    </wps:Input>
  </wps>DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput>
      <ows:Identifier>bounds</ows:Identifier>
    </wps:RawDataOutput>
  </wps:ResponseForm>
</wps:Execute>
```

使用postman模拟结果

```
<?xml version="1.0" encoding="UTF-8"?>
<ows:BoundingBox xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ows="http://www.opengis.net/ows/1.1" crs="EPSG:4326">
  <ows:LowerCorner>-124.73142200000001 24.955967</ows:LowerCorner>
  <ows:UpperCorner>-66.969849 49.371735</ows:UpperCorner>
</ows:BoundingBox>
```

2、重新投影Geoserver上的图层

WPS request builder

Step by step WPS request builder.

Choose process

vec:Reproject

Reprojects features into a supplied coordinate reference system. Can also force a feature collection to have a g

Process inputs

features* - SimpleFeatureCollection

The feature collection that will be reprojected

VECTOR_LAYER tiger:giant_polygon

forcedCRS - CoordinateReferenceSystem

Coordinate reference system to use for input feature collection (overrides native one)

EPSG:4326 查找... EPSG:WGS 84...

targetCRS - CoordinateReferenceSystem

Target coordinate reference system to use for reprojection

EPSG:2326 查找... EPSG:Hong Kong 1980 Grid System...

Process outputs

result* - SimpleFeatureCollection

Input feature collection

☒ Generate application/json

Authentication

☐ Authenticate (will run the request as anonymous otherwise)

Execute process

Generate XML from process inputs/outputs

```
{
  "type": "FeatureCollection",
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:2326"
    }
  },
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [
              [
                [836496.1844, -1.16514017382E7],
                [836368.3267, 8352802.7341],
                [836368.3267, 8352802.7341],
                [836496.1844, -1.16514017382E7],
                [836496.1844, -1.16514017382E7]
              ]
            ]
          ]
        ]
      },
      "properties": {
        "id": "giant_polygon.1"
      }
    }
  ]
}
```

```
<?xml version="1.0" encoding="UTF-8"?><wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
  <ows:Identifier>vec:Reproject</ows:Identifier>
  <wps>DataInputs>
    <wps:Input>
      <ows:Identifier>features</ows:Identifier>
      <wps:Reference mimeType="text/xml" xlink:href="http://geoserver/wfs"
method="POST">
        <wps:Body>
          <wfs:GetFeature service="WFS" version="1.0.0" outputFormat="GML2"
xmlns:tiger="http://www.census.gov">
            <wfs:Query typeName="tiger:giant_polygon"/>
          </wfs:Body>
        </wps:Reference>
      </wps:Input>
    </wps>DataInputs>
  </wps:Execute>
```



```

        </wfs:GetFeature>
    </wps:Body>
</wps:Reference>
</wps:Input>
<wps:Input>
    <ows:Identifier>forcedCRS</ows:Identifier>
    <wps>Data>
        <wps:LiteralData>EPSG:4326</wps:LiteralData>
    </wps>Data>
</wps:Input>
<wps:Input>
    <ows:Identifier>targetCRS</ows:Identifier>
    <wps>Data>
        <wps:LiteralData>EPSG:2326</wps:LiteralData>
    </wps>Data>
</wps:Input>
</wps>DataInputs>
<wps:ResponseForm>
    <wps:RawDataOutput mimeType="application/json">
        <ows:Identifier>result</ows:Identifier>
    </wps:RawDataOutput>
</wps:ResponseForm>
</wps:Execute>

```

使用postman模拟结果

```

{
  "type": "FeatureCollection",
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:2326"
    }
  },
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [
              [
                836496.1844,
                -11651401.7382
              ],
              [
                836368.3267,
                8352802.7341
              ],
              [
                836368.3267,
                8352802.7341
              ],
              [
                836496.1844,
                -11651401.7382
              ]
            ]
          ]
        ]
      }
    }
  ]
}

```

```
],
[
    836496.1844,
    -11651401.7382
]
]
]
],
"properties": {},
"id": "giant_polygon.1"
}
]
}
```

3、生成具有地理标定的单元格规则网格

Step by step WPS request builder.

Choose process

gs:Grid

Generates a georeferenced regular grid of cells. Output contains the attributes: cell - the cell polygon; id - a unique identifier; centerX and centerY - the ordinates of the cell center. (WPS DescribeProcess)

Process inputs

bounds* - ReferencedEnvelope

Bounds of the grid

最小 X	最小 Y	最大 X	最大 Y
0	0	200	200

坐标参考系

EPSG:4326 查找... EPSG:WGS 84...

width* - Double

Width of a cell (in units of the grid CRS)

20

height - Double

Height of a cell (in units of the grid CRS). Only for rectangular grid, defaults to equal width.

20

vertexSpacing - Double

Distance between vertices along cell sides (in units of the grid CRS)

5

mode* - GridMode

Type of grid to be generated. Specifies shape of cells in grid.

Rectangular

Process outputs

result* - SimpleFeatureCollection

Generated grid cells as features

☒ Generate application/json

Authentication

☐ Authenticate (will run the request as anonymous otherwise)

```

{"type": "FeatureCollection", "features": [{"type": "Feature", "geometry":
{"type": "Polygon", "coordinates": [[[0.0, 0.0], [0.0, 20], [20, 20], [20, 0.0],
[0.0, 0.0]]}], "properties":
{"id": 0, "centerX": 10.0, "centerY": 10.0, "id": "grid.0"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[20, 0.0],
[20, 20], [40, 20], [40, 0.0], [20, 0.0]]}], "properties":
{"id": 1, "centerX": 30.0, "centerY": 10.0, "id": "grid.1"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[40, 0.0],
[40, 20], [60, 20], [60, 0.0], [40, 0.0]]}], "properties":
{"id": 2, "centerX": 50.0, "centerY": 10.0, "id": "grid.2"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[60, 0.0],
[60, 20], [80, 20], [80, 0.0], [60, 0.0]]}], "properties":
{"id": 3, "centerX": 70.0, "centerY": 10.0, "id": "grid.3"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[80, 0.0],
[80, 20], [100, 20], [100, 0.0], [80, 0.0]]}], "properties":
{"id": 4, "centerX": 90.0, "centerY": 10.0, "id": "grid.4"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[100, 0.0],
[100, 20], [120, 20], [120, 0.0], [100, 0.0]]}], "properties":
{"id": 5, "centerX": 110.0, "centerY": 10.0, "id": "grid.5"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[120, 0.0],
[120, 20], [140, 20], [140, 0.0], [120, 0.0]]}], "properties":
{"id": 6, "centerX": 130.0, "centerY": 10.0, "id": "grid.6"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[140, 0.0],
[140, 20], [160, 20], [160, 0.0], [140, 0.0]]}], "properties":
{"id": 7, "centerX": 150.0, "centerY": 10.0, "id": "grid.7"},
{"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[160, 0.0],

```

```

<?xml version="1.0" encoding="UTF-8"?><wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
  <ows:Identifier>gs:Grid</ows:Identifier>
  <wps>DataInputs>
    <wps:Input>
      <ows:Identifier>bounds</ows:Identifier>
      <wps>Data>
        <wps:BoundingBoxData crs="EPSG:4326" dimensions="2">
          <ows:LowerCorner>0.0 0.0</ows:LowerCorner>
          <ows:UpperCorner>200.0 200.0</ows:UpperCorner>
        </wps:BoundingBoxData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>width</ows:Identifier>
      <wps>Data>
        <wps:LiteralData>20</wps:LiteralData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>height</ows:Identifier>
      <wps>Data>
        <wps:LiteralData>20</wps:LiteralData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>vertexSpacing</ows:Identifier>
      <wps>Data>
        <wps:LiteralData>5</wps:LiteralData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>mode</ows:Identifier>

```

```
<wps:Data>
  <wps:LiteralData>Rectangular</wps:LiteralData>
</wps:Data>
</wps:Input>
</wps:DataInputs>
<wps:ResponseForm>
  <wps:RawDataOutput mimeType="application/json">
    <ows:Identifier>result</ows:Identifier>
  </wps:RawDataOutput>
</wps:ResponseForm>
</wps:Execute>
```

postman模拟结果

```
{
  "type": "Feature",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [
          0,
          0
        ],
        [
          0,
          20
        ],
        [
          20,
          20
        ],
        [
          20,
          0
        ],
        [
          0,
          0
        ]
      ]
    ]
  },
  "properties": {
    "id": 0,
    "centerX": 10,
    "centerY": 10
  },
  "id": "grid.0"
}
```

